DISK DRIVE 1541-II

Guida per l'Utente

1541

UNITÀ SINGOLA A FLOPPY DISK

GUIDA PER L'UTENTE

Commodore Italiana S.p.A.
aprile 1984

INFORMAZIONI PER L'UTILIZZATORE

ATTENZIONE: questo apparecchio è conforme alle caratteristiche dei dispositivi di calcolo appartenenti alla classe B, sottoparte J della sezione 15 delle norme FCC.
A questo apparecchio possono essere collegati solamente altri apparecchi aventi le stesse caratteristiche. In caso contrario è possibile il verificarsi di interferenze nella ricezione radio e TV.

Questo apparecchio genera ed usa segnali a radiofrequenza e, se non correttamente installato secondo le norme dettate dal costruttore, può causare disturbi alla ricezione radio e TV. Malgrado esso appartenga al gruppo BJ15 delle norme FCC, ciò non garantisce l'immunità da interferenze in particolari situazioni. In un caso di tale tipo, che si può facilmente verificare accendendo e spegnendo l'apparecchio, l'utilizzatore può intervenire:

- orientando l'antenna TV
- spostando l'apparecchio
- allontanando l'apparecchio dal ricevitore
- alimentando i due apparecchi con due differenti prese di rete, appartenenti a due diverse maglie di alimentazione.

Se necessario, l'utilizzatore deve rivolgersi al rivenditore o ad un tecnico qualificato per adottare le misure necessarie ad evitare l'inconveniente.

Le informazioni contenute in questo manuale sono state accuratamente controllate. Tuttavia, non si assume alcuna responsabilità per eventuali errori presenti. Il contenuto del manuale ha unicamente scopo informativo ed è soggetto a variazioni senza preavviso.

INDICE

1. DESCRIZIONE GENERALE

Introduzione

Benvenuti tra la crescente schiera di utilizzatori della più veloce, efficiente e semplice da usare unità di memoria di massa per il vostro computer VIC 20 o Commodore 64: l'unità a floppy disk 1541. Questo manuale ci permetterà di trarre da essa le migliori prestazioni, indipendentemente dalle vostre conoscenze di informatica.

Se non siete degli esperti, leggendo i primi capitoli potrete conoscere le regole fondamentali per installare ed usare le vostra unità a floppy disk. Via via che migliorerete le vostre conoscenze, potrete passare ai capitoli successivi che vi permetteranno di conoscerne vita e miracoli.

Se invece siete dei professionisti, il manuale vi servirà come riferimento per collegare l'unità al vostro computer e per utilizzarla al meglio delle prestazioni.

Comunque, indipendentemente dall'esperienza che avete, l'unità a floppy disk 1541 migliorerà notevolmente le possibilità el vostro sistema.

Prima di iniziare, è bene fare alcune premesse. Questo manuale costituisce un riferimento esclusivamente per l'unità a floppy disk VIC-1541; pertanto, se avete bisogno di informazioni sulla programmazione, dovete consultare i manuali di istruzioni in dotazione al vostro computer VIC 20 o Commodore 64.

È inoltre bene che impariate il BASIC ed i comandi necessari per usare l'unità a floppy, anche se in questo manuale vi forniremo passo-passo le sequenze necessarie per compiere ogni operazione. Se invece pansate di usare l'unità solamente con del software già esistente, è anche presente nel manuale una breve sezione che vi permetterà di fare ciò in modo semplice e rapido.

Ed ora, incominciamo con le informazioni generali.

I comandi per l'unità a floppy sono presentati nel manuale con un livello crescente di complessità. Partendo dal Capitolo 3, conoscerete prima i comandi per salvare un programma su floppy e per richiamarlo in memoria. Nel Capitolo 4 vedrete come o comandi vengono inviati all'unità e quali sono i comandi per manipolare i file sul floppy.

Il Capitolo 5 vi insegna a lavorare con file sequenziali che sono simili a quelli su nastro, anche se la loro gestione è molto più veloce. Nel Capitolo 6 vedrete come si fa a lavorare sui file casuali, ad accedere a qualunque posizione del disco ed in che modo è organizzato il floppy stesso, in tracce ed in blocchi. Il Capitolo 7 descrive i file relativi che costituiscono il miglior metodo per la memorizzazione di archivi, specialmente quando sono usati in unione a file sequenziali.

Nel Capitolo 8 potrete conoscere i metodi di programmazione in linguaggio macchina del controllore dell'unità a floppy, mentre l'ultimo capitolo, il numero 9, vi insegnerà a cambiare l'indirizzo della vostra unità a floppy, sia intervenendo direttamente sull'hardware, che per mezzo del software.

Ricordatevi comunque che non è necessario imparare tutto ciò che è descritto in questo manuale in un sol colpo. I primi 4 capitoli sono più che sufficienti per incominciare ad usare l'unità, e con il contenuto dei due capitoli successivi potete compiere quasi tutte le operazioni che considerate. Una sempre maggior confidenza con la vostra unità a floppy vi ripagherà in mille modi, ad esempio con una maggior velocità, affidabilità e flessibilità nelle vostre operazioni di trattamento dei dati.

Caratteristiche dell'unità a floppy disk

Questa unità di memoria a floppy disk permette di memorizzare fino a 144 programmi e/o file di dati su un unico floppy disk da 5" 1/4, singola faccia, singola densità, per un massimo di 174 kbyte.

Nell'unità è contenuta l'elettronica relativa al controllore ed al sistema operativo (firmware), per un totale di 16 kbyte su ROM e 2 kbyte su RAM. Grazie alla presenza di questi circuiti, l'unità a floppy disk VIC-1541 deve essere considerata una periferica intelligente. Ciò significa che funziona senza aver bisogno della memoria del computer VIC 20 o Commodore 64. L'unità si avvale di un sistema detto "pipeline" che la rende capace i ricevere ed eseguire i comandi ad essa diretti, mentre il computer è impegnato in altri compiti. In tal modo, le prestazioni generali del sistema risultano drasticamente migliorate.

I floppy disk registrati con questa unità risultano perfettamente compatibili sia in lettura, che in scrittura, con le unità Commodore 4040 e 2031.
Inoltre, con la 1541 è anche possibile leggere programmi creati sulle precedenti unità 2040.

L'unità a floppy disk 1541 è dotata di un bus seriale duale, ideato appositamente dalla Commodore. I segnali su questo bus ricordano quelli sul bus parallelo IEEE-488 utilizzato dai computer PET della Commodore, tranne per il fatto che, in questo caso viene utilizzato un solo conduttore, anziché otto. Le due porte sul retro dell'unità permettono a più di una periferica di condividere contemporaneamente il bus seriale. Ciò è possibile collegando i dispositivi in cascata, ognuno al precedente ed al successivo. In tal modo, è possibile collegare contemporaneamente al computer fino a cinque unità a floppy ed una stampante.

Tabella 1.1 Caratteristiche dell'unità singola a floppy disk VIC 1540/1541

CAPACITÀ DI MEMORIA

Capacità totale	174.848 byte per floppy
Sequenziali	168.656 byte per floppy
Relativi	167.132 byte per floppy 65.535 record per file
Voci nel direttorio	144 per floppy
Settori per ogni traccia	da 17 a 21
Byte per ogni settore	256
Tracce totali	35 per floppy
Blocchi totali	683 (blocchi liberi 664)

CIRCUITI INTEGRATI UTILIZZATI

6502	microprocessore
6522 (2)	I/O, temporizzatori interni
Buffer	
2114 (4)	2 k RAM

CARATTERISTICHE FISICHE

Geometriche

Altezza	77 mm
Larghezza	184 mm
Profondità	256 mm

Elettriche

Tensione di alimentazione	100, 120, 220 o 240V CA
Frequenza di alimentazione	50/60 Hz
Potenza assorbita	25W

CARATTERISTICHE DEL SUPPORTO

Floppy disk	5" 1/4, singola faccia, singola densità

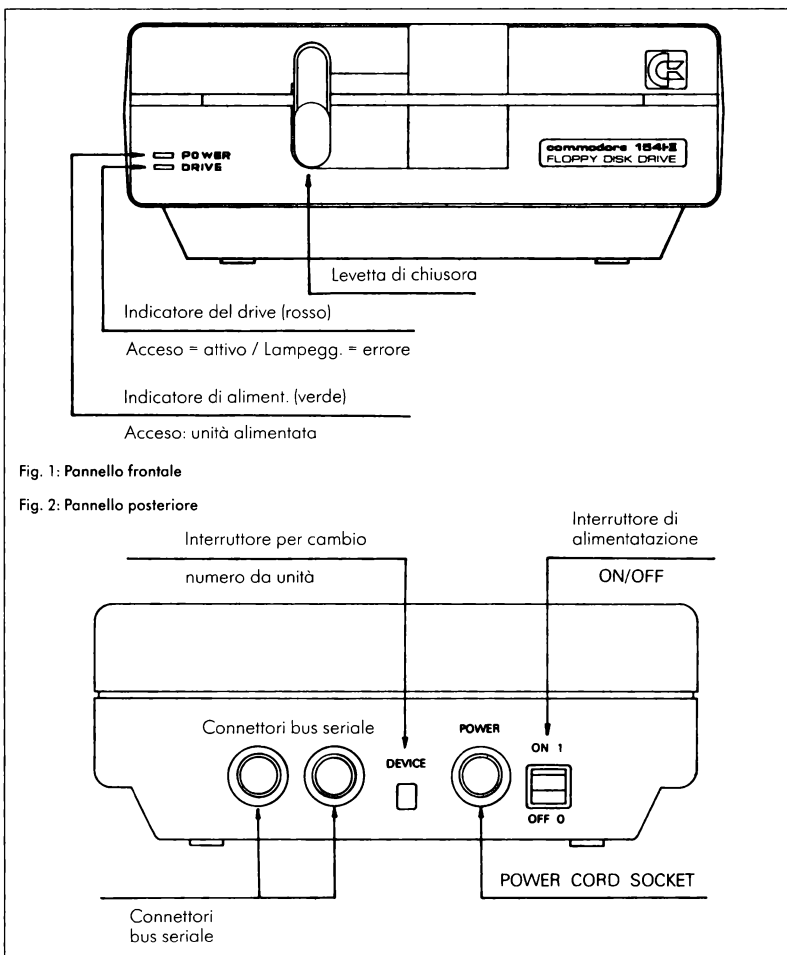
2. **DISIMBALLAGGIO E COLLEGAMENTI**

Contenuto della confezione

Nel contenitore troverete, assieme all'unità a floppy disk 1541, il cavo di alimentazione di colore grigio, il cavo di interfaccia seriale nero, il presente manuale ed un floppy disk dimostrativo. Il cavo di alimentazione ha ad un estremo un connettore per il collegamento sul pannello posteriore dell'unità, ed all'altro estremo una spina a tre conduttori (con terra) per il collegamento alla rete di alimentazione. Il cavo nero di interfaccia ha alle estremità due connettori identici a 6 piedini (norme DIN) per il collegamento al VIC 20, al Commodore 64, oppure ad un'altra unità a floppy.

Vi raccomandiamo di non effettuare alcun collegamento prima di aver letto attentamente questo capitolo.

Fig. 1: **Pannello frontale**

Fig. 2: **Pannello posteriore**

Collegamento dei cavi

La prima operazione consiste nel prendere il cavo di alimentazione e collegarlo correttamente sul retro dell'unità a floppy, facendo riferimento alla figura 2.2. Dopo di ciò inserite la spina all'altro capo del cavo nella presa di rete.

Se, a questo punto, sentite dei rumori provenire dall'unità a floppy, spegnetela per mezzo dell'interruttore di alimentazione sul pannello posteriore.
Non collegate assolutamente altri cavi all'unità quando la stessa è accesa.

Collegate quindi il cavo di interfaccia a uno dei due connettori appositi sul pannello posteriore dell'unità.
Spegnete il vostro computer e collegate al relativo connettore di interfaccia sul pannello posteriore l'altro capo del cavo. I collegamenti sono così terminati.

Se volete collegare anche una stampante od un'altra unità a floppy disk, collegate i dispositivi in cascata utilizzando la porta seriale dell'unità a floppy, come mostrato in figura 2.3. Se dovete usare più unità a floppy contemporaneamente, fate riferimento al capitolo 9. Se è la prima volta che usate le unità a floppy, incominciate con una sola, in modo da prendere confidenza con la stessa.

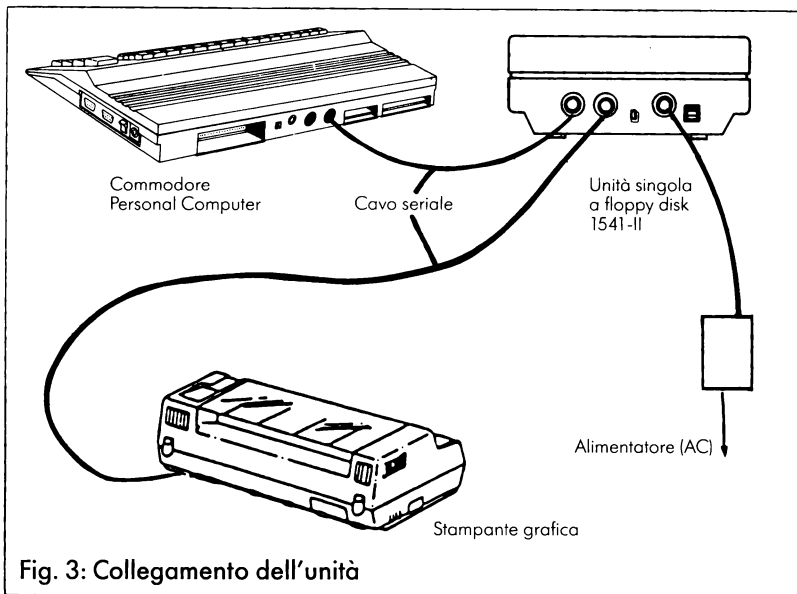



Fig. 3: Collegamento dell'unità

Accensione

Dopo aver collegato correttamente i vari dispositivi, è giunto il momento di fornire ad essi l'alimentazione.
È molto importante, per il corretto funzionamento del sistema, che seguiate l'ordine di accensione descritto.
Il computer deve essere sempre acceso per ultimo. Se si osserva questa semplice regola, il sistema dovrebbe funzionare alla perfezione.

Pertanto, assicuratevi innanzitutto che non vi siano floppy disk inseriti nell'unità.

Accendete tutte le periferiche e quindi il computer: le periferiche partono con le loro sequenze di inizializzazione. Si avvia il motore della stampante, la testina della stessa si muove fino a circa metà della riga e quindi ritona in posizione di partenza. Nell'unità a floppy 1541, si accende la spia rossa, mentre sullo schermo TV si forma l'immagine di inzializzazione.

Quando tutte le spie sull'unità a floppy smettono di lampeggiare, l'unità è pronta.

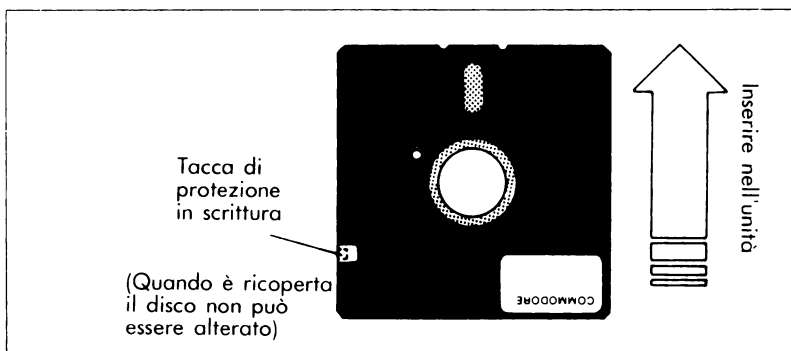



Fig. 4. Inserimento del floppy

Inserimento del floppy disk nell'unità

Per aprire lo sportello di protezione dell'unità basta tirare leggermente in alto la linguetta; la protezione si apre e l'eventuale floppy presente viene spinto fuori. Prendete il nuovo floppy ed inseritelo con la finestra di lettura rivolta verso l'alto e la tacca di protezione a sinistra (nel floppy dimostrativo è ricoperta con nastro adesivo) facendo riferimento alla figura 2.4.

Premete il floppy delicatamente fino a farlo penetrare completamente nella fessura; sentirete uno scatto ed il floppy non sarà più spinto fuori. Chiudete lo sportello premendolo leggermente verso il basso fino a bloccaggio. Siete ora pronti a lavorare con il floppy.

Ricordatevi sempre di rimuovere il floppy disk prima di spegnere l'apparecchio. **Non estraete. assolutamente il floppy quando la spia rossa è accesa poiché correte il rischio di perdere irrimediabilmente i dati in esso contenuti.**

Uso dell'unità con il VIC 20 od il Commodore 64

L'unità a floppy disk 1541 può essere usata sia con un computer VIC 20, che con un Commodore 64.
Tuttavia, i due computer hanno valori diversi della velocità di trasferimento dei dati per cui occorre effettuare una commutazione via software per selezionare la velocità adatta al computer in uso. L'unità viene consegnata pronta per essere usata con un Commodore 64. Per utilizzarla con un VIC 20 occorre dare il seguente comando dopo aver acceso l'apparecchio:

OPEN 15,8,15, "UI-": CLOSE 15

Per utilizzare nuovamente l'unità con il Commodore 64, occorre invece dare il comando:

OPEN 15,8,15, "UI+": CLOSE 15

Nel capitolo 4 verranno date ulteriori informazioni su questo tipo di comandi, mentre una dettagliata descrizione dei comandi U (user) è data al Capitolo 8.

3. TRATTAMENTO DEI PROGRAMMI

CARICAMENTO DI PROGRAMMI GIÀ ESISTENTI SU FLOPPY

Se vi interessa lavorare solamente con programmi già esistenti su floppy, ecco tutto ciò che dovete fare:

inserire con attenzione il floppy nell'unità, tenendo l'etichetta dello stesso in alto e vicina a voi e la tacca di protezione, che deve essere ricoperta con nastro adesivo, a sinistra. Chiudete lo sportello dell'unità, scrivete da tastiera il comando LOAD "NOME DEL PROGRAMMA", 8 e premete il tasto **RETURN**.
L'unità a disco si avvia e sullo schermo appaiono successivamente le scritte:

che indicano rispettivamente "ricerca in atto del programma desiderato" "caricamento del programma trovato nella memoria centrale del computer" e "pronto".
A questo punto il programma è pronto per essere usato e non resta che dare il comando RUN.

LOAD

I comandi BASIC usati con l'unità a floppy sono gli stessi validi per l'unità a cassette Commodore DatacassetteTM.
Inoltre, vi sono alcuni altri comandi previsti solo per l'unità a floppy.
Una prima differenza consiste nel fatto che, mentre l'unità a cassette è una memoria di massa sequenziale e quindi è possibile dare il comando LOAD senza specificare il nome del programma, con l'effetto di caricare il primo programma presente sulla cassetta, nel caso del floppy vi sono numerosi programmi con lo stesso grado di accessibilità, per cui occorre specificare il nome del programma desiderato.
Una seconda differenza consiste nell'obbligo di specificare il numero di dispositivo dell'unità a floppy. Se ciò non viene fatto il computer presuppone che il programma cercato si trovi su cassetta ed effettua la ricerca nell'unità a cassetta.

FORMATO DEL COMANDO **LOAD**

LOAD nome del programma$, numero del dispositivo, numero del comando

Il nome del programma è una stringa cioè un nome tra virgolette, oppure il valore di una assegnata variabile di stringa. Ecco alcuni esempi di nomi validi: "HELLO", "PROGRAM#1", A$, NAMES.

Il "numero del dispositivo" viene predisposto in fabbrica sul circuito stampato pari ad 8. Se state lavorando non più di una unità a floppy, fate riferimento al Capitolo 8 per cambiare questo indirizzo. Nel seguito, supporremo che tale valore sia pari ad 8.

Il "numero del comando" è opzionale. Se non è indicato o è posto pari a zero, il programma viene caricato normalmente a partire dall'inizio della memoria disponibile per i programmi BASIC. Se tale numero è pari ad 1, il programma sarà caricato esattamente **nelle stesse locazioni di memoria in cui era stato scritto in origine.** Nel caso di computer con differenti configurazioni della memoria, come ad esempio VIC 20 con 5 k, 8 k o più, l'inizio della memoria per i programmi BASIC si trova in locazioni differenti. Il valore 0 provoca il caricamento normale dei programmi BASIC; il valore 1 è utilizzato essenzialmante per caricare programmi in linguaggio macchina, per set di caratteri e per altre funzioni dipendenti dalla locazione in memoria.

> **NOTA:** Per definire il numero del dispositivo, del comando e le stringhe, potete usare anche delle variabili, purché le stesse siano state precedentemente definite nel programma.

Direttorio del floppy disk

L'unità a nastro DatacassetteTM è un dispositivo sequenziale di memoria di massa. Essa può quindi leggere e scrivere solamente dall'inizio alla fine del nastro, senza effettuare salti e senza la possibilità di accedere a file in modo selettivo.

L'unità a floppy disk è invece una memoria di massa ad accesso casuale. La testina di lettura/scrittura può raggiungere direttamente qualunque punto del floppy ed accedere a singoli blocchi di dati, ognuno contenente fino a 256 byte di informazione. Su ogni floppy vi sono 683 blocchi.

Per fortuna, non è compito vostro lavorare sui singoli blocchi. Sul floppy esiste infatti un programma speciale chiamato DOS (Disk Operating System) che ricorda per voi la successione dei blocchi, per mezzo del BAM (Block Availability Map) e del direttorio.

Il file BAM è un elenco dei 683 blocchi presenti sul floppy, caricato sullo stesso, a metà tra il centro e la circonferenza esterna. Ogni volta che salvate sul floppy un programma (SAVE) o chiudete un file di dati (CLOSE) il BAM viene aggiornato automaticamente con la nuova situazione dei blocchi.

Il file, direttorio è un elenco di tutti i programmi e dei file contenuti sul floppy, ed è fisicamente situato vicino al BAM. Il direttorio può contenere fino a 144 gruppi di dati, ognuno consistente in una serie di informazioni come il nome ed il tipo del file, l'elenco dei blocchi che lo compongono ed il blocco di inizio dello stesso.

Anche il direttorio è aggiornato automaticamente ogni volta che salvate su floppy un programma (SAVE) o aprite un file di dati (OPEN).
Fate attenzione: il BAM non viene aggiornato finché il file non viene chiuso (CLOSE), anche se il direttorio è già stato aggiornato, per cui **se un file dati non viene chiuso correttamente (CLOSE) correte seri pericoli di perderlo completamente.**

Il direttorio può essere caricato (LOAD) nel vostro computer come un qualunque programma BASIC. Ponete il floppy nel 1541 ed introducete il seguente comando:

LOAD "$", 8

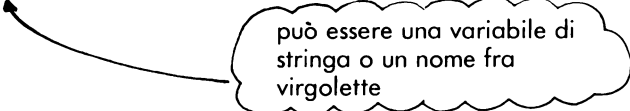
Sullo schermo appaiono le seguenti scritte:

```
SEARCHING FOR $
FOUND $
LOADING
READY.
```

A questo punto, il direttorio del floppy è stato caricato nella memoria centrale del vostro computer. Battete LIST e lo vedrete visualizzato sullo schermo.
Se invece lo volete stampare con la stampante di sistema, battete i seguenti comandi (si suppone che il numero di dispositivo della stampante sia pari a 4):

OPEN 4, 8, 4: CMD: LIST

> **NOTA:** Quando si usa il comando CMD, il file deve essere chiuso con il comando PRINT#4: CLOSE 4. Per una spiegazione più dettagliata, fate riferimento al manuale della stampante VIC 1525/1515.

Se invece volete leggere il direttorio **senza** caricarlo nella memoria del computer, fate riferimento al programma di supporto del DOS, più avanti in questo capitolo. Inoltre, per leggere il direttorio mentre vi trovate in un programma BASIC, fate riferimento all'istruzione GET#, nel Capitolo 5.

Ricerca approssimata su floppy disk.

Quando usate l'unità a cassette, potete caricare in memoria (LOAD) un qualunque programma il cui nome inizia con una o più lettere, senza scrivere tutte le lettere del nome. Ad esempio LOAD "T" significa "carica il primo programma che trovi, il cui nome inizia con la lettera T" e LOAD "HELLO" significa "carica il primo programma che trovi, il cui nome inizia con la parola "HELLO", come ad esempio il programma "HELLO THERE".

Quando si utilizza l'unità a floppy, questa possibilità, chiamata "pattern matching" è costituita da un carattere speciale che sostituisce, nel nome del file cercato, la o le lettere non conosciute. Ponendo un asterisco (*) come terminatore di una qualunque serie di caratteri, ordinate all'unità di cercare un file il cui nome che incomincia con quella serie di caratteri.

FORMATO PER IL "PATTERN MATCHING"

LOAD nome$+ "*", 8

In altre parole, se volete caricare (LOAD) il primo programma presente nel direttorio del floppy il cui nome inizia con T, dovete scrivere LOAD "T*", 8.

Se il nome è costituito solamente da "*", verrà caricato (LOAD) l'ultimo programma a cui avete avuto accesso sul floppy. Se nessun programma è stato ancora caricato, verrà caricato il primo del direttorio.

Conoscerete certamente il significato del jolly (wild card) a poker: esso può sostituire qualunque altra carta.
Sul 1541 voi potete usare il carattere (?) come jolly per sostituire una lettera qualunque durante la ricerca di un file su floppy. Il nome del programma cercato è confrontato con quello presente sul floppy. Il nome del programma cercato è confrontato con quello presente sul floppy, carattere per carattere, tranne in corrispondenza del (?).

Ad esempio, con il comando LOAD "T?NT", 8 i programmi che potete caricare sono TINT, TENT, e così via.

Quando caricate (LOAD) il direttorio del floppy, potete usare * e ? per fare delle ricerche selettive. Se ad esempio date il comando LOAD "$0: TEST", apparirà nel direttorio solo il programma TEST, ovviamente se presente sul floppy. Con LOAD "$0: T*" otterrete l'elenco di tutti i programmi che iniziano con T e con LOAD "$0: T?ST" quello di tutti i programmi con nome di 4 lettere in cui la prima è una T, le ultime due sono ST e la seconda una lettera qualunque.
Infine. LOAD "$0: T?ST*" elencherà tutti i nomi di lunghezza qualunque con le lettere T, S, T rispettivamente in prima, terza e quarta posizione.

SAVE

Per salvare un programma su floppy, basta scrivere il numero del dispositivo dopo il nome del file. Analogamente al comando SAVE per l'unità a nastro, il numero del dispositivo può essere seguito dal numero del comando per impedire la riallocazione automatica in seguito ad un comando LOAD (fate riferimento al paragrafo relativo al comando LOAD).

FORMATO DEL COMANDO **SAVE**

SAVE nome del programma$, numero del dispositivo, numero del comando

Fate riferimento al comando LOAD (pagina 11) (*) per una spiegazione dei parametri numero del dispositivo e numero del comando.

Quando dite all'unità a floppy di salvare un programma (SAVE), il DOS deve effettuare numerosi passi. Innanzitutto, esso cerca nel direttorio per vedere se esiste già un programma con quel nome e se esiste un campo ancora disponibile. Quindi verifica nel BAM se esistono abbastanza blocchi liberi per memorizzare quel programma. Se ognuna di queste verifiche dà esito positivo, il programma viene salvato; in caso contrario inizia a lampeggiare la spia di errore.

SAVE e REPLACE

Se un programma esiste già sul floppy, è spesso necessario effettuare delle modifiche e quindi rimemorizzarlo sullo stesso. In questo caso è comodo cancellare la vecchia versione e salvare la versione revisionata.

Se il primo carattere del nome del programma è costituito dal simbolo "@" a seguito da uno 0 e da (:), il DOS sostituisce la vecchia versione del programma con quel nome con il programma presente nella memoria centrale del computer. L'unità a floppy verifica nel direttorio la presenza della vecchia versione del programma, quindi prende nota che tale nome è stato cancellato e successivamente crea il nuovo campo con lo stesso nome. Infine, il programma viene memorizzato normalmente.

FORMATO DEL COMANDO **SAVE CON REPLACE**

SAVE "@0": "+nome $, numero del dispositivo, numero del comando

Ad esempio, per un file chiamato TEST, il comando ha la forma: SAVE "@0: TEST", 8.

La presenza di 0: è necessaria per assicurare la compatibilità con le altre unità a floppy Commodore che hanno più di un drive. In tal caso, vengono usati i numeri 0 ed 1 per specificare quale dei due drive presenti si desidera utilizzare.

VERIFY

Il comando VERIFY permette di controllare un programma presente in memoria per confronto con uno presente sul floppy. Con il comando VERIFY occorre specificare il numero del dispositivo. Il computer confronta i due programmi byte per byte, compresi i "line links" che possono essere differenti per diverse configurazioni della memoria. Ad esempio, se un programma è stato salvato (SAVE) su floppy partendo da un VIC 20 con 5 k di memoria e quindi viene caricato (LOAD) su un computer con 8 k, esso non può essere verificato (VERIFY) correttamente perché i link puntano a differenti locazioni della memoria.

FORMATO DEL COMANDO **VERIFY**

VERIFY nome $, numero del dispositivo.

Programma DOS SUPPORT

Sul floppy dimostrativo è presente un programma chiamato DOS SUPPORT che vi permette di usare più facilmente numerosi comandi dell'unità a floppy (in modo diverso per il VIC 20 o per il Commodore 64). Vi basta caricare (LOAD) e lanciare (RUN) il programma. Esso si inizializza automaticamente d altrettanto automaticamente si cancella quando avete terminato di usarlo. In tal modo avete a disposizione alcune centinaia di byte in meno nella memoria principale, ma in compenso potete inviare i vostri comandi all'unità a floppy molto più semplicemente.

Ad esempio, il tasto "/" viene ridefinito come comando LOAD. Vi basta battere "/" prima del nome del file ed esso viene immediatamente caricato in memoria. Non è quindi necessario l'uso del comando LOAD con il formato descritto in precedenza.

Anche i tasti '@" e ">" sono usati per inviare comandi all'unità a disco. Battendo @$ (oppure >$) il direttorio del disco viene visualizzato sullo schermo senza doverlo caricare (LOAD) in memoria.
Questi tasti sostituiscono anche PRINT# per inviare i comandi elencati nel capitolo successivo (vedi Capitolo 4).

Per leggere il canale di errore del floppy (quando la spia rossa è lampeggiante) basta premere i tasti @ o > e quindi RETURN. Sullo schermo viene visualizzato il messaggio completo di errore comprendente numero, testo e numeri della traccia e dei blocchi.

4. COMANDI DELL'UNITÀ

OPEN e PRINT#

Fino a questo momento abbiamo visto alcuni metodi abbastanza semplici per lavorare con l'unità a floppy. Per comunicare con essa in modo più completo, occorre utilizzare le istruzioni BASIC OPEN e PRINT# (troverete maggiori informazioni a tale proposito sui manuali in dotazione ai vostri computer VIC 20 e Commodore 64). Forse conoscete già queste istruzioni per averle usate con i file di dati su cassetta, dove OPEN crea il file e PRINT# lo riempie con i dati. Queste istruzioni possono essere usate nello stesso modo con i floppy, come vedrete nel capitolo successivo, ma possono essere anche usate per inizializzare un canale di comando per scambiare informazioni tra il computer e l'unità a floppy.

FORMATO DELL'ISTRUZIONE **OPEN**

OPEN numero del file, numero del dispositivo, numero del canale, testo $

Il numero del file può essere un numero qualunque compreso tra 1 e 255 e permette di identificare, all'interno del programa, il file a cui si è avuto accesso. È bene però evitare numeri maggiori di 127 perché essi fanno generare all'istruzione PRINT# un a capo dopo il carattere di ritorno. Questi numeri sono utili con stampati non standard.

Il numero di dispositivo per l'unità a floppy vale normalmente 8.

Il numero di canale può essere qualunque numero compreso tra 2 e 15. Esso fa riferimento al canale usato per comunicare con l'unità a floppy, mentre i canali 0 ed 1 sono riservati al sistema operativo per i comandi SAVE e LOAD. I canali da 2 a 14 possono essere usati per i dati ed il canale 15 è il **canale di comando.**

Text$ è una stringa che viene stampata (PRINT) sul file come un'istruzione PRINT#. Ciò è utile per inviare un comando singolo al canale.

Il comando PRINT# lavora esattamente come l'istruzione PRINT, tranne per il fatto che i dati sono inviati ad un dispositivo (l'unità a floppy in questo caso) anziché allo schermo. Quando usato con un canale dati, PRINT# invia le informazioni ad un buffer nell'unità a disco, dal quale poi va sul floppy. Quando invece PRINT# è usato con un canale comando, i comandi sono inviati all'unità a disco.

FORMATO PER INVIARE I COMANDI ALL'UNITÀ A FLOPPY:

OPEN 15, 8, 15, comando$

oppure

PRINT# 15, comando$

NEW

Questo comando è necessario quando si usa un floppy per la prima volta. Il comando NEW cancella totalmente il floppy, fissa i limitatori di blocco e crea il direttorio ed il BAM. Il comando NEW può anche essere usato per pulire il direttorio di un floppy già formattato. Questo metodo è più rapido della riformattazione di tutto il floppy.

FORMAT DEL COMANDO **NEW** PER FORMATTARE IL FLOPPY

PRINT# 15, "NEWϕ: nome, id"   (NUMERO DEL DRIVE)

oppure, in forma abbreviata:

PRINT# 15, "Nϕ: nome, id"

FORMATO DEL COMANDO **NEW** PER PULIRE IL DIRETTORIO

PRINT# 15, "Nϕ: nome"

Il nome entra nel direttorio come nome di tutto il floppy ed è l'unico che appare quando si LISTA lo stesso. Il codice ID è costituito da 2 caratteri qualunque che vengono posti non solo nel direttorio, ma anche in tutti i blocchi del floppy. In tal modo, se per caso sostituiste il floppy durante la scrittura, l'unità si accorgerebbe di ciò verificando ID.

COPY

Questo comando vi permette di copiare qualunque programma o file con l'unità a floppy. Esso non vi abilita a copiare da un'unità (tranne nel caso di unità con due drive come il 4040) ma permette di duplicare un programma con un nome di diverso sulla stessa unità.

FORMATO DEL COMANDO **COPY**

PRINT# 15, "COPYϕ: nuovo file:=ϕ: vecchio file" (NUMERO DELL'UNITÀ)

oppure, in forma abbreviata:

PRINT# 15, "Cϕ: nuovo file=ϕ: vecchio file"

Il comando COPY può anche essere usato per combinare più file in uno solo, sul floppy.

FORMATO DEL COMANDO **COPY** PER COMBINARE FILE

PRINT# 15, "C0: nuovo file=0: vecchio file 1,0: vecchio file 2,ϕ: vecchio file 3,ϕ: vecchio file 4"

ESEMPI DEL COMANDO **COPY**

PRINT# 15, "C0: BACKUP=0: ORIGINAL"

PRINT# 15, "C0: MASTERFILE=0: NAME, 0: ADDRESS, 0: PHONES"

RENAME

Questo comando vi permette di cambiare il nome di un file che si trova già nel direttorio. Si tratta di un'operazione rapida poiché si interviene solo sul nome.

FORMATO DEL COMANDO **RENAME**

PRINT# 15, "RENAMEϕ: nuovo nome=vecchio nome" (NUMERO DELL'UNITÀ)

oppure, in forma abbreviata:

PRINT# 15, "Rϕ: nuovo nome=vecchio nome"

ESEMPIO DEL COMANDO **RENAME**

PRINT# 15, "R0: MYRA= MYRON"

Il comando RENAME non funziona con file che sono aperti (OPEN)

SCRATCH

Questo comando vi permette di cancellare file e programmi dal floppy, rendendo lo spazio da essi occupato disponibile per nuove operazioni. Potete cancellare i programmi uno alla volta o in gruppi utilizzando i caratteri speciali visti per la ricerca approssimata (* e ?).

FORMATO DEL COMANDO **SCRATCH**

PRINT# 15, "SCRATCH0: nome" NUMERI DELL'UNITÀ A FLOPPY

oppure, in forma abbreviata:

PRINT# 15, "S0: nome"

Se verificate il canale di errore dopo un'operazione di SCRATCH (vedi sotto), il numero normalmente utilizzato per la traccia vi dirà quanti file sono stati cancellati. Il comando funziona così: se ad esempio, nel direttorio sono presenti i programmi KNOW e GANW e voi scrivete PRINT# 15, "S0: ?N?W", **entrambi i programmi verranno cancellati.**
Nello stesso modo, se sono presenti i programmi TEST, TRAIN, TRUCK e TAIL e voi scrivete PRINT# 15, "S0: T*", **cancellate tutti i 4 programmi.**

INITIALIZE

Talvolta, una condizione di errore che si verifica nell'unità a disco vi impedisce determinate operazioni che desiderate effettuare. Il comando INITIALIZE riporta l'unità nelle stesse condizioni che ha all'accensione, a questo punto può essere necessario riconfigurare l'unità per il computer che state usando.

FORMATO PER IL COMANDO **INITIALIZE**

PRINT# 15, "INITIALIZE"

oppure, in forma abbreviata:

PRINT# 15, "I"

VALIDATE

Dopo che il floppy è stato usato per un po' di tempo, il direttorio può perdere l'ordinamento. Quando numerosi programmi vengono salvati (SAVE) e quindi cancellati (SCRATCH), rimangono sul floppy dei blocchi di minime dimensioni, che non possono errere riutilizzati perché troppo piccoli. Il comando VALIDATE vi permette di riordinare il floppy in modo da ottenere il massimo spazio utilizzabile.

Inoltre, possono esservi dei file di dati aperti (OPEN) e non chiusi correttamente (CLOSE) VALIDATE raccoglie tutti questi blocchi e li rende nuovamente disponibili , poiché i file di dati non sono a quel punto più utilizzabili.

Esiste una controindicazione nell'uso di questo comando. In presenza di file casuali (vedi Capitolo 6), i blocchi allocati vengono disallocati. Pertanto VALIDATE non deve essere con floppy che contengono file casuali.

FORMATO PER IL COMANDO **VALIDATE**

PRINT# 15, "VALIDATE"

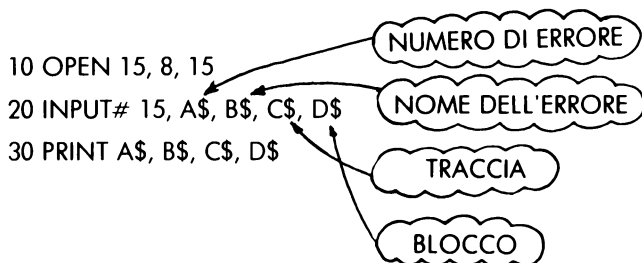
oppure, in forma abbreviata:

PRINT# 15, "V"

DUPLICATE

Questo comando è legato ai sistemi operativi delle unità duali come il 4040 e permette di copiare un floppy completo da un drive all'altro. Non ha effetto su un'unità a floppy singola.

Lettura del canale di errore

Senza il programma DOS SUPPORT non potete leggere il canale di errore del floppy, poiché dovete usare il comando INPUT# che non funziona fuori da un programma. Ecco una semplicissima routine BASIC che permette di leggere il canale di errore:

Ogniqualvolta effettuate un'operazione INPUT# dal canale comando, potete leggere fino a 4 variabili che decrivono la condizione di errore. La prima, la terza e la quarta sono valori numerici e possono essere assegnate (INPUT) a variabili numeriche.

La prima fornisce il numero dell'errore e se vale 0 significa "assenza di errori"; la seconda costituisce la descrizione dell'errore; la terza e la quarta danno rispettivamente il numero della traccia e del blocco (settore) in cui si è verificato l'errore.

Errori nella traccia 18 sono relativi al BAM ed al direttorio. Ad esempio un READ ERROR nel blocco 0 della traccia 18 indica che il floppy non è mai stato formattato.

CLOSE

È di estrema importanza che vi ricordiate di chiudere (CLOSE) i file che avete terminato di usare. Tale operazione permette al DOS di allocare correttamente i blocchi nel BAM e di terminare l'introduzione nel direttorio. **Se non chiudete (CLOSE) il file, tutti i vostri dati verranno perduti.**

FORMATO PER L'ISTRUZIONE **CLOSE**

CLOSE numero del file

Dovete porre un'estrema attenzione a non chiudere (CLOSE) il canale di errore (canale numero 15) prima di aver chiuso i canali dati. **Il canale di errore deve essere aperto per primo (OPEN) e chiuso per ultimo (CLOSE).** Solo in tal modo lavorerete con i vostri programmi senza problemi.

Se chiudete il canale di errore mentre altri file sono aperti, l'unità a disco li chiude per voi, ma il BASIC pensa che siano ancora aperti e vi permette di provare a scrivere in essi.

> **NOTA:** Se il vostro programma BASIC vi porta ad una condizione di errore, tutti i file vengono chiusi in BASIC, **senza** che gli stessi siano chiusi sull'unità a floppy. **Questa situazione è estremamente pericolosa.**

Dovete immediatamente scrivere l'istruzione OPEN 15, 8, 15, "I" per inizializzare nuovamente l'unità a floppy e salvare i file stessi.

5. **FILE SEQUENZIALI**

OPEN

I file sequenziali sul floppy sono perfettamente equivalenti a quelli a cassetta, tranne per il fatto che il loro uso è più veloce. Essi sono limitati dalla loro natura sequenziale e ciò significa che debbono essere letti in sequenza dall'inizio alla fine. I dati sono trasferiti sul floppy byte per byte, attraverso un buffer e sono visti identici dall'unità a floppy. Ciò significa che file sequenziali, file di programmi e file utente lavorano allo stesso modo. In effetti, l'unica differenza è che solo i programmi possono essere caricati in memoria (LOAD); stesso discorso vale per il direttorio, anche se in sola lettura. I file relativi, invece, si comportano in modo differente.

FORMATO PER L'APERTURA DI **FILE SEQUENZIALI**

OPEN numero di file, numero del dispositivo, numero del canale, "0: nome, tipo, direzione".

Il numero del file ha lo stesso significato visto in precedenza per l'istruzione OPEN e viene usato dal programma per fare riferimento a quel file particolare. Il numero del dispositivo vale al solito 8. Il numero di canale è un canale dati, da 2 a 14, ed è comodo usare lo stesso numero per canale e file. Il nome rappresenta il nome del file, senza caratteri speciali se state creando un file di scrittura. Il tipo può essere uno qualunque tra quelli elencati nella tabella sotto riportata, indicato anche in forma abbreviata con la prima lettera. La direzione può essere READ o WRITE ed è permessa l'abbreviazione.

TIPO DI FILE	**SIGNIFICATO**
PRG	Programma
SEQ	Sequenziale
USR	Utente
REL	Relativo (non sopportato dal BASIC 2.0)

ESEMPI DI APERTURA DI FILE SEQUENZIALI

```
OPEN 2, 8, 2 "0: DATA, $, W"
OPEN 8, 8, 8, "0: Program, P, R"
OPEN A, B, C, "0:" +A$+ "U, W"
```

NOME DEL FILE

TIPO

READ/WRITE

Se il file esiste già, è possibile usare l'opzione **REPLACE** simile al comando SAVE e REPLACE decritto nel Capitolo 3. Basta aggiungere il carattere @0: prima del nome del file nell'istruzione OPEN.

```
OPEN 2, 8, 2 "@0: DATA, S, W"
```

PRINT # e INPUT #

Il comando PRINT # funziona **esattamente** come PRINT, tranne per il fatto che l'uscita viene inviata nuovamente all'unità floppy. In particolare, tutte le possibilità di formattazione del comando PRINT (anche per quanti riguarda la punteggiatura) sono perfettamente valide anche per PRINT #, per cui occorre prestare particolare attenzione all'introdurre i dati nel file.

FORMATO DI SCRITTURA DI UN FILE CON PRINT #

PRINT # numero del file, lista dei dati

Il numero del file è quello che è stato usato quando il file è stato creato con il comando OPEN.

La lista dei dati è la stessa del comando PRINT, cioé un elenco di variabili e/o testo tra virgolette. Tuttavia, dovete porre particolare attenzione quando scrivete i dati perché la loro lettura successiva non comporti problemi.

Con il comando PRINT #, se usate (,) per separare i dati su una linea, essi saranno separati da alcuni spazi vuoti, come se fossero visualizzati sullo schermo. Se invece usate (;) i dati saranno posizionati l'uno accanto all'altro.

Per capire meglio che cosa accade, ecco lo schema di un file sequenziale cerato con l'istruzione OPEN 5, 8, 5, "0: TEST, S, W".

	eof															
char	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	. . .

dove eof significa "end of file".

Le sequenze di dati vengono inserite byte per byte, compresi gli spazi.

Ad esempio, inizializziamo alcune variabili con le istruzioni A$="HELLO": B$="ALL": C$="BYE". Ecco un'immagine del file dopo l'istruzione PRINT # 5, A$; B$; C$:

	H	E	L	L	O	A	L	L	B	Y	E	CR	eof
char	1	2	3	4	5	6	7	8	9	10	11	12	13

CR significa **codice CHR$** di 13, cioè **il ritorno carrello,** che viene inviato alla fine di ogni comando PRINT o PRINT#, a meno che non sia presente (,) o (;) al termine della riga.

Nota: non lasciate spazi tra PRINT e # e non tentate di abbreviare il comando come ?#. Fate riferimento alla "Guida per l'uso" per le corrette abbreviazioni.

FORMATO PER L'ISTRUZIONE **INPUT#**

INPUT# numero del file, lista di variabili

Qaundo usate INPUT# per leggere i dati uno di seguito all'altro, non c'è nessun elemento per indicare che non si tratta di un'unica stringa lunga.

Occorre quindi qualche cosa che funzioni da separatore. I caratteri usati a tale scopo sono CR, (,) o (;). CR può essere facilmente aggiunto scrivendo semplicemente una variabile per ogni linea nell'istruzione PRINT#, ed il sistema pone automaticamente CR. La serie PRINT# 5, A$: PRINT# 5, B$: PRINT# 5, C$ pone un CR dopo ogni variabile permettendo la corretta separazione per un'istruzione tipo PRINT# 5, A$, B$, C$. Lo stesso effetto è ottenuto anche con la sequenza: Z$=",": PRINT# 5, A$ Z$ B$ Z$ C$ in meno spazio. Il file ottenuto dopo questa linea si presenta così:

	H	E	L	L	O	,	A	L	L	,	B	Y	E	CR	eof
char	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Ponendo delle virgole tra le variabili si consuma molto più spazio sul disco. Ad esempio PRINT# 5, A$, B$ assume il seguente aspetto:

	H	E	L	L	O						A	L	L		CR	eof
char	1	2	3	4	5	6	7	8	9	10	11	12	13	14	. . .	23	24

Potete vedere che in questo modo molto spazio va perduto.

In definitiva, fate molta attenzione quando usate PRINT# e non avrete problemi quando vorrete leggere i vostri dati.

I dati numerici scritti nei file assumono la forma di una stringa come se si avesse operato su di essi con una funzione STR$, prima di farli stampare. Il primo carattere sarà uno spazio bianco se il numero è positivo ed un segno (—) se il numero è negativo. Segue quindi il numero, e l'ultimo carattere a destra rappresenta il cursore. Questo formato fornisce sufficienti informazioni per l'istruzione INPUT# per leggere i valori numerici separati, anche se sono stati scritti senza altri separatori speciali. Sussiste comunque una perdita di spazio con due caratteri non utilizzati quando il numero è positivo.

Ecco l'immagine del file dopo aver impostato PRINT# 5, 1; 3; 5; 7:

		1	⇒		3	⇒		5	⇒		7	⇒	CR	eof	
char	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

L'appendice B contiene un programma dimostrativo sull'uso di file sequenziali su floppy disk.

GET#

Il comando GET# ritrova i dati su floppy, un carattere dopo l'altro.

FORMATO PER L'ISTRUZIONE **GET#**

GET# numero del file, lista di variabili

I dati vengono ripresi byte per byte, compreso il carattere CR, le virgole e gli altri separatori. È meglio con GET# usare variabili di stringa poiché, se quando è richiesto un valore numerico viene fornita una stringa, il BASIC invia un messaggio di errore, ma non viceversa.

ESEMPI DELL'ISTRUZIONE GET#

```
GET# 5, A$

GET# A, B$, C$, D$

GET$ 5, A
```

L'istruzione GET# è molto utile quando si esaminano file con contenuto non noto, come ad esempio un file che potrebbe essere stato danneggiato da un programma sperimentale. È più sicuro di INPUT# perché c'è un limite al numero di caratteri permessi tra i separatori delle variabili di INPUT. Con GET# potete ricevere ogni carattere ed esaminare anche i separatori nello stesso modo degli altri.

Ecco un esempio di programma che vi permette di esaminare qualunque file su floppy.

```
 10 INPUT "FILE NAME"; F$
 20 INPUT "FILE TYPE"; T$
 30 T$=LEFT$(T$,1)
 40 IF T$ <> "S" THEN IF T$ <> "P" THEN  IF T$ <> "U" THEN 20
 45 OPEN 15, 8, 15
 50 OPEN 5, 8, 5, "0:"+F$+ ","+T$+ ",R"
 60 GOSUB 200
 70 GET# 5, A$
 80 IF ST <> 0 THEN PRINT ST: STOP
 90 PRINT ASC(A$+CHR$(0));
100 GOTO 70
200 INPUT# 15, A$, B$, C$, D$
210 IF VAL (A$) >0 THEN PRINT A$, B$, C$, D$: STOP
220 RETURN
```

In caso di carattere nullo essendo in READ IN. In caso contrario si ottiene errore con la funzione ASC.

Lettura del direttorio

Il direttorio del disco può essere letto nello stesso modo di un file sequenziale. Basta usare $ come nome del file, e OPEN 5, 8, 5, "$"; l'istruzione GET# permette quindi di esaminare il direttorio. Il formato è identico a quello di un programma; la dimensione del file è data dai numeri di linea ed i nomi sono memorizzati come stringhe di caratteri tra virgolette.

Ecco un programma che vi permette di leggere il direttorio del floppy:

```
 10 OPEN1,8,2,"$
 20 GET#1,A$,A$,A$,A$
 30 T$(0)="Del":T$(1)="SEQ":T$(2)="PRG":T$(3)="USR":T$(4)="REL"
 40 J=17:GOSUB500   <-- NOME DEL DEL DISCO
 50 N$=B$
 60 J=2             <-- ID
 70 GOSUB500
 80 I$=B$
 90 J=2             <-- SISTEMA OPERATIVO
100 GOSUB500
110 O$=B$
120 FORL=1TO73      <-- DÀ IL RESTO DEL BLOCCO
130 GET#1,A$,A$,A$
140 NEXT
150 GET#1, A$,A$,A$,A$,A$
160 PRINTCHR$(147) "Disk name: "N$, "ID: "I$,"OS: "O$
161 PRINT"Length","Type","Name"
165 FORP=1TO8
170 GET#1,T$,A$,A$
175 IFSTTHENCLOSE1:END
180 IFT$=""THENT$=CHR$(128)
190 J=15            <-- NOME DEL FILE
200 GOSUB500
210 N$=B$
220 GET#1,A$,A$,A$,A$;A$,A$,A$,A$,A$,A$,L$,H$   <-- DÀ LA LUNGHEZZA DEL FILE
225 L=ASC(L$+CHR$(0))+256*ASC(H$+CHR$(0):IFL=0THEN250
230 PRINTL,T$(ASC(T$)-128),N$
250 IFP<8THENGET#1,A$,A$
260 NEXTP: GOTO165
500 B$=""
510 FORL=0TOJ
520 GET#1,A$
530 IFA$<>CHR$(96)THENIFA$<>CHR$(160)THENB$=B$+A$   COSTITUISCE UNA SUBROUTINE DI STRINGA
540 NEXT
550 RETURN
```

Tabella 5.1: FORMATO BAM DEL 1540/1541

Traccia 18, settore 0.		
BYTE	CONTENUTO	DEFINIZIONE
0,1	18,01	Traccia e settore del 1° blocco del direttorio
2	65	Codice ASCII del carattere A che indica il formato del 4040
3	0	Flag nullo per usi futuri del DOS
4 - 143		* bit map dei blocchi disponibili 1-35
*1 = blocco disponibile 0 = blocco non disponibile (ogni bit rappresenta un blocco)		

Tabella 5.2: TITOLO DEL DIRETTORIO DEL 1540/1541

Traccia 18, Settore 0		
BYTE	CONTENUTO	DEFINIZIONE
144-161		Nome del disco con spazi shiftati
162-163		ID del floppy
164	160	Spazio shiftato
165-166	50,65	Codice ASCII di 2A che rappresenta la versione ed il formato del DOS
166-167	160	Spazi shiftati
171-255	0	Nulli, non usati
Nota: codici ASCII possono comparire nelle locazioni da 180 a 191 di alcuni floppy.		

FORMATO DEL DIRETTORIO

Traccia 18, settore 1 per 4040	
Traccia 39, settore 1 per 8050	
BYTE	DEFINIZIONE
0,1	Traccia e settore del blocco successivo del direttorio
2-31	* Campo file 1
34-63	* Campo file 2
66-95	* Campo file 3
98-127	* Campo file 4
130-159	* Campo file 5
162-191	* Campo file 6
194-223	* Campo file 7
226-255	* Campo file 8

* STRUTTURA DI UN SINGOLO CAMPO DEL DIRETTORIO

BYTE	CONTENUTO	DEFINIZIONE
0	128+tipo	Tipo del file in OR logico con $80 per indicare correttamente la chiusura del file. TIPI: 0 = DELeted (cancellato) 1 = SEQuenziale 2 = PROGramma 3 = USER 4 = RELativo
1-2		Traccia e settore del 1° blocco di dati
3-18		Nome del file con spazi shiftati
19-20		Solo per file relativi: traccia e settore di inizio
21		Solo per file relativi: dimensioni del record
22-25		Non usato
26-27		Traccia e settore di sostituzione del file con OPEN @
28-29		Numero dei blocchi nel file: primo ed ultimo byte

Tabella 5.4: FORMATO DI FILE SEQUENZIALE

BYTE	DEFINIZIONE
0,1	Traccia e settore del successivo blocco dati
2-256	254 byte di dati con ritorno carrello e terminatori del record

Tabella 5.5: FORMATO DI FILE PROGRAMMA

BYTE	DEFINIZIONE
0,1	Traccia e settore del successivo blocco
2-256	254 byte di programma in formato CBM. La fine del file è individuata da 3 byte nulli.

6. FILE CASUALI

I file sequenziali funzionano bene quando state lavorando con flussi continui di dati, ma alcune situazioni richiedono una organizzazione diversa. Ad esempio, in presenza di una lista di spedizioni di notevoli dimensioni, la ricerca del record relativo ad una persona è bene che non sia fatta semplicemente scorrendo tutta la lista. Per tale motivo è necessario avere la possibilità di **accesso casuale,** in modo da poter accedere direttamente ad un record del floppy senza dover leggere l'intero file.

Sull'unità a floppy Commodore esistono due metodi per l'accesso casuale. I file relativi, di cui si parlerà nel prossimo capitolo, sono più comodi per il trattamento di dati anche se in questo capitolo sono stati usati file casuali, specialmente con programmi in linguaggio macchina.

I file casuali sono costituiti da singoli blocchi di 256 byte di dati. Come detto nel Capitolo 1, sono presenti su un floppy vergine 683 blocchi, dei quali 664 a disposizione dell'utilizzatore. In pratica, ogni blocco di dati significa una traccia ed il settore con lo stesso nome.

Il floppy è diviso in tracce che hanno la forma di cerchi concentrici sulla superficie del disco. Su ogni floppy vi sono tracce numerate in modo che la 1 è quella più esterna e la 35 è quella più vicina al centro. La traccia 18 è occupata dal direttorio e dal BAM ed il DOS inserisce le informazioni partendo dal centro verso l'esterno.

Ogni traccia è a sua volta suddivisa in settori. Poiché le tracce più esterne sono più lunghe, contengono un numero maggiore di settori. Le tracce esterne hanno infatti 21 settori, mentre quelle interne, solo 17. Nella tabella che segue sono elencati i numeri dei settori per ogni traccia.

Tabella 6.1: FORMATO DELLE TRACCE E DEI BLOCCHI

NUMERO DI TRACCIA	SETTORI/TRACCIA	TOTALE DEI SETTORI
da 1 a 17	da 0 a 20	21
da 18 a 24	da 0 a 18	19
da 25 a 30	da 0 a 17	18
da 31 a 35	da 0 a 16	17

Il DOS contiene i comandi per accedere direttamente in lettura ed in scrittura a qualunque traccia e settore del floppy. Vi sono anche dei comandi per verficare i blocchi (tracce e settori) disponibili e per marcare quelli usati.

Questi comandi sono inviati per mezzo del canale comando (numero 15) e dicono all'unità di trattare i dati. I dati possono essere letti successivamente per mezzo dei canali dati aperti.

Apertura di un canale dati per accesso casuale

Quando lavorate con file casuali, avete bisogno di due canali aperti sul disco, uno per i comandi ed uno per i dati. Il canale comando è aperto come canale 15 (OPEN), nello stesso modo delle altre situazioni già incontrate. IL canale dati per i file ad accesso casuale viene aperto (OPEN) ponendo come nome del file il simbolo (#).

FORMATO DELL'ISTRUZIONE **OPEN PER FILE DATI AD ACCESSO CASUALE**

OPEN numero del file, numero del dispositivo, numero del canale, "#"
oppure:
OPEN numero del file, numero del dispositivo, numero del canale, "# numero del buffer"

ESEMPI DI APERTURA DI UN CANALE DATI PER ACCESSO CASUALE

BLOCK-READ

FORMATO DEL COMANDO **BLOCK-READ**

PRINT# numero del file, "BLOCK-READ:" canale, drive, traccia, blocco
oppure, in forma abbreviata:
PRINT# numero del file, "B-R:" canale, drive, traccia, blocco

Questo comando sposta un blocco di dati dal floppy nel canale selezionato. Dopo aver effettuato questa operazione, è possibile leggere le informazioni con le istruzioni INPUT# e GET#.

ESEMPIO DI PROGRAMMA PER LEGGERE IL BLOCCO 2 DALLA TRACCIA 18 (e memorizza i contenuti in B$)

```
10 OPEN 15, 8, 15
20 OPEN 5, 8, 5, "#"
30 PRINT# 15, "B-R:" 5, 0, 18, 2
40 B$=""
50 FOR L=0 TO 255
60 GET# 5, A$
70 IF ST=0 THEN B$= B$+ A$: NEXT L
80 PRINT "FINISHED"
90 CLOSE 5: CLOSE 15
```

BLOCK-WRITE

Il comando BLOCK-WRITE ha la funzione opposta a BLOCK-READ e permette, dopo aver caricato in un buffer delle informazioni, di scaricare le stesse sul floppy nel modo desiderato.

FORMATO PER IL COMANDO **BLOCK-WRITE**

PRINT# numero del file, "BLOCK-WRITE:" drive, canale, traccia, blocco
oppure, in forma abbreviata:
PRINT# numero del file, "B-W:" drive, canale, traccia, blocco

Quando un dato viene inserito nel buffer, un puntatore del DOS tiene traccia del numero di caratteri. Quando usate l'istruzione BLOCK-WRITE, anche il puntatore viene registrato sul floppy. Ciò spiega la presenza della verifica ST nella linea 70 del programma visto sopra: ST diventa diverso da 0 quando cercate di leggere oltre il puntatore di fine file (eof) all'interno del record.

ESEMPIO DI PROGRAMMA PER SCRIVERE DEI DATI NEL SETTORE 1 della traccia 1

```
10 OPEN 15, 8, 15
20 OPEN 5, 8, 5, "#"
30 FOR L=1 to 50
40 PRINT#5, "TEST"
50 NEXT
60 PRINT# 15, "B-W:" 5, 0, 1, 1
70 CLOSE 5: CLOSE 15
```

BLOCK-ALLOCATE

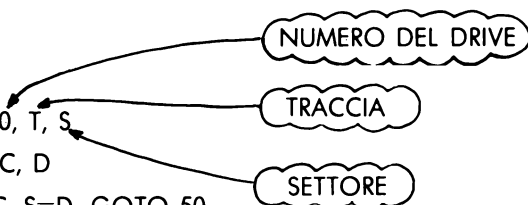
Allo scopo di usare con sicurezza file casuali con file normali, i vostri programmi debbono verificare nel BAM la presenza di blocchi disponibili e quindi aggiornare il BAM per indicare che li state usando. Dopo che il BAM è stato aggiornato, i vostri file casuali sono stati salvati, **a meno che non usiate il comando VALIDATE** (vedi Capitolo 3).

FORMATO DEL COMANDO **BLOCK-ALLOCATE**

PRINT# numero del file, "BLOCK-ALLOCATE:" drive, traccia, blocco

Come si può sapere quali blocchi sono liberi? Se provate a scrivere su un blocco non disponibile, il DOS vi invia il messaggio di errore 65, NO BLOCK, e vi dice quali sono i primi numeri di traccia e di blocco disponibili. Pertanto, ogni volta che cercate di scrivere in un blocco sul floppy, dovete prima provare di allocarlo. Se il blocco non risulta disponibile, leggete sul canale di errore il numero del primo blocco successivo disponibile e quindi effettuate l'allocazione dello stesso.

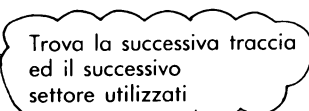
ESEMPIO DELLA PROCEDURA PER ALLOCARE IL BLOCCO

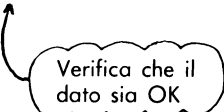
```
10 OPEN 15, 8, 15
20 OPEN 5, 8, 5, "#
30 PRINT# 5, "DATA"
40 T=1: S=1
50 PRINT#15, "B-A:" 0, T, S
60 INPUT#15, A, B$, C, D
70 IF A=65 THEN T=C: S=D: GOTO 50
80 PRINT# 15, "B-W:" 5, 0, T, S
```

BLOCK-FREE

Il comando BLOCK-FREE ha la funzione opposta a BLOCK-ALLOCATION poiché libera un blocco. Esso può essere considerato un lontano parente di SCRATCH per i file, poiché in realtà non cancella i dati dal floppy, ma semplicemente rende i blocchi accessibili sul BAM.

FORMATO PER IL COMANDO **BOCK-FREE**

PRINT# numero del file, "BLOCK-FREE:" drive, traccia, blocco
oppure, in forma abbreviata:
PRINT# numero del file, "B-F:" drive, traccia, blocco

Utilizzazione dei file casuali

L'unico problema che esiste lavorando con i file casuali consiste nell'impossibilità di tenere una traccia dei blocchi che avete usato. Non potete infatti conoscere il numero di blocco usato sul BAM da un altro blocco e non potete sapere se contiene un file casuale, una parte di programma, una sequenza o dei file relativi. Per tenere una traccia, il metodo più comune consiste nel costruire un file sequenziale parallelo ad ogni file causale: un file di tale tipo contiene l'elenco delle locazioni di record, traccia e blocco. Ciò significa avere per ogni file casuale 3 canali aperti sul floppy: uno per i comandi, uno per i dati casuali ed uno per i dati sequenziali. Ciò inoltre significa che dovrete riempire contemporaneamente due buffer con i dati.

ESEMPIO DI PROGRAMMA PER SCRIVERE IN 10 BLOCCHI AD ACCESSO CASUALE CREANDO IL RELATIVO FILE SEQUENZIALE:

```
 10 OPEN 15, 8, 15
 20 OPEN 5, 8, 5, "#
 30 OPEN 4, 8, 4, "@0:KEYS,S,W"
 40 A$= "RECORD CONTENTS#"
 50 FOR R=1 TO 10
 70 PRINT# 5, A$ "," R
 90 T=1:B=1
100 PRINT# 15, "B-A:" 0, T, B
110 INPUT# 15, A, B$, C, D
120 IF A=65 THEN T=C:B=D: GOTO 100
130 PRINT# 15, "B-W:" 5, 0, T, B
140 PRINT# 4, T ;B
150 NEXT R
160 CLOSE 4: CLOSE 5: CLOSE 15
```

ESEMPIO DI PROGRAMMA PER LEGGERE IN 10 BLOCCHI AD ACCESSO CASUALE ED IL RELATIVO FILE SEQUENZIALE

```
 10 OPEN 15, 8, 15
 20 OPEN 5, 8, 5, "#"
 30 OPEN 4, 8, 4, "KEYS,S,R"
 40 FOR R=1 TO 10
 50 INPUT# 4, T, S
 60 PRINT# 15, "B-R:" 5, 0, T, S
 80 INPUT# 5, A$, X
 90 IF A$<>"Record Contents#" OR X <> R THEN STOP
110 PRINT# 15, "B-F:" 0, T, S
120 NEXT R
130 CLOSE 4: CLOSE 5
140 PRINT# 15, "S0:KEYS"
150 CLOSE 15
```

Trova la successiva traccia ed il successivo settore utilizzati

Verifica che il dato sia OK

Puntatore del buffer (Comando BUFFER-POINTER)

Il puntatore del buffer tiene traccia di dove sono stati scritti gli ultimi dati e da dove si deve iniziare a leggere i dati successivi. Cambiando la locazione del puntatore nel buffer, potete accedere in modo casuale a singoli byte all'interno di un blocco. In questo modo potete dividere ogni blocco in più record.

Ad esempio, supponiamo che stiate lavorando su una lista di spedizione e che le informazioni come nome, indirizzo, eccetera, occupino al massimo 64 caratteri. Potete dividere ogni blocco in 4 record separati e, conoscendo i numeri di record, di settore e di traccia, potete accedere ai singoli record.

FORMATO PER IL COMANDO **BUFFER-POINTER**

PRINT# numero di file, "BUFFER-POINTER:" canale, locazione
oppure, in forma abbreviata:
PRINT# numero di file, "B-P:" canale, locazione

ESEMPIO DI POSIZIONAMENTO DEL PUNTATORE SUL 64ESIMO CARATTERE DEL BUFFER

PRINT# 15, "B-P:" 5, 64

Ecco le versioni dei programmi di scrittura e lettura ad accesso casuale già visti, modificati per operare su record all'interno dei blocchi.

ESEMPIO DI PROGRAMMA PER SCRIVERE IN 10 BLOCCHI AD ACCESSO CASUALE, OGNUNO FORMATO DA 4 RECORD

```
 10 OPEN 15, 8, 15
 20 OPEN 5, 8, 5, "#
 30 OPEN 4, 8, 4, "KEYS,S,W"
 40 A$= "RECORD CONTENTS#"
 50 FOR R=1 TO 10
 60 FRO L=1 TO 4
 70 PRINT# 15, "B-P:" 5; (L-1)* 64
 80 PRINT# 5, A$ "," L
 90 NEXT L
100 T=1:B=1
110 PRINT# 15, "B-A:" O; T; B
120 IN PUT# 15, A, B$, C, D
130 IF A=65 THEN T=C:B=D: GOTO 110
140 PRINT# 15, "B-W:" 5; 0; T; B
150 PRINT# 4, T ;B
160 NEXT R
170 CLOSE 4: CLOSE 5: CLOSE 15
```

Posizionamento a 0, 64, 128 o 192

Trova traccia e settore disponibili

ESEMPIO DI PROGRAMMA PER LEGGERE IN 10 BLOCCHI AD ACCESSO CASUALE, OGNUNO FORMATO DA 4 RECORD

```
 10 OPEN 15, 8, 15
 20 OPEN 5, 8, 5, "#
 30 OPEN 4, 8, 4, "KEYS,S,R"
 40 FOR R=1 TO 10
 50 INPUT# 4, T, S
```

```
 60 PRINT# 15, "B-R:" 5; 0; T; S
 70 FOR L=1 TO 4
 80 PRINT# 15, "B-P:" 5; (L-1)* 64
 85 INPUT# 5, A$, X
 90 IF A$<>"Record Contents#" OR X=L THEN STOP
100 NEXT L
110 PRINT# 15, "B-F:" 0; T; S
120 NEXT R
130 CLOSE 4: CLOSE 5
140 PRINT# 15, "S0:KEYS"
150 CLOSE 15
```

USER1 ed USER2

I comandi USER sono generalmente previsti per lavorare in linguaggio macchina (fate riferimento, per la maggior parte di essi, al prossimo capitolo). I comandi USER1 ed USER2 costituiscono delle versioni particolari di BLOCK-READ e BLOCK-WRITE, con una sostanziale differenza: il modo di lavorare con il puntatore del buffer.

Il comando BLOCK-READ legge fino a 256 caratteri, ma termina la lettura quando il puntatore di buffer memorizzato con il blocco indica che lo stesso è terminato.

FORMATO DEL COMANDO USER1

PRINT# numero del file, "U1:" canale, drive, traccia, blocco
oppure:
PRINT# numero del file, "UA:" canale, drive, traccia, blocco

Per questo comando non esiste alcuna differenza tra gli indicatori U1 ed UA.

Il comando BLOCK-WRITE scrive il contenuto del buffer nel blocco con il valore del puntatore di buffer. USER2 scrive nel buffer senza toccare il valore del puntatore già memorizzato in quel blocco. Ciò è utile quando un blocco deve essere letto con BLOCK-READ, aggiornato con BUFFER-POINTER e PRINT# e quindi riscritto con USER2.

FORMATO PER IL COMANDO USER2

PRINT# numero del file, "U2:" canale, drive, traccia, blocco
oppure:
PRINT# numero del file, "UB:" canale, drive, traccia, blocco

Per un esempio più complesso dell'uso del comando, fate riferimento all'appendice B.

7. **FILE RELATIVI**

I file relativi vi permettono di operare esattamente sui dati che desiderate nell'ambito di un file. Essi sono più comodi per il trattamento dei dati perché vi permettono di strutturare i vostri file in record ed in campi all'interno dei record.

Il DOS ricerca le tracce ed i settori usati e permette anche di sovrapporre dei record da un blocco al successivo. È possibile fare ciò perché vengono fissati dei **side sectors,** una serie di puntatori per l'inizio di ogni record. Ogni side sector può puntare fino a 120 record e possono esservi 6 side sector in un file. Ogni file può contenere fino a 720 settori ed ogni record 254 caratteri. Ciò significa che si può avere un solo file che occupa tutto il floppy disk.

Creazione di un file relativo

Quando si vuole usare un file relativo per la prima volta, occorre prima crearlo con l'istruzione OPEN. L'opzione REPLACE **non funziona** con questo tipo di file. I file relativi possono essere espansi, letti e scritti.

FORMATO PER L'ISTRUZIONE OPEN PER CREARE FILE RELATIVI

OPEN numero del file, numero del dispositivo, numero del canale, "nome,L, "+CHR$ (lunghezza del record)

ESEMPI DELL'ISTRUZIONE OPEN PER CREARE FILE RELATIVI

Tabella 7.1: FORMATO DI FILE RELATIVI

BLOCCO DATI	
BYTE	DEFINIZIONE
0,1	Traccia e settore del successivo blocco di dati
2-256	254 byte di dati. I record vuoti contengono FF (tutti uni binari) nel primo byte, seguiti da 00 (tutti zeri binari) fino a completamento del record. I record parzialmente occupati sono riempiti con (00).
BLOCCO SIDE SECTOR	
BYTE	DEFINIZIONE
0,1	Traccia e settore del successivo blocco di dati
2	Numero del side sector (0-5)
3	Lunghezza del record
4-5	Traccia e settore del primo side sector (numero 0)
6-7	Traccia e settore del secondo side sector (numero 1)
8-9	Traccia e settore del terzo side sector (numero 2)
10-11	Traccia e settore del quarto side sector (numero 3)
12-13	Traccia e settore del quinto side sector (numero 4)
14-15	Traccia e settore del sesto side sector (numero 5)
16-256	Puntatori di traccia e settore ai 120 blocchi dati

Durante l'esecuzione, il DOS verifica se il file che si vuole creare già esiste ed in tale evenienza non accade nulla. L'unico modo per cancellare un file relativo già esistente è per mezzo del comando SCRATCH (vedi Capitolo 4), perché **non è possibile** usare l'opzione REPLACE.

Utilizzazione dei file relativi

FORMATO PER APRIRE UN FILE RELATIVO GIÀ ESISTENTE

OPEN numero del file, numero del dispositivo, numero del canale, "nome"

In questo caso, il DOS sa automaticamente che si tratta di un file relativo. Questa sintassi e quella vista nella sezione precedente permettono sia la scrittura che la lettura del file.

Per leggere o scrivere, dovete, **prima di effettuare l'operazione,** posizionare il puntatore sul record che vi interessa.

FORMATO PER IL COMANDO **POSITION**

PRINT# numero del file, "P" CHR$ (numero del canale) CHR$ (primo numero del record) CHR$ (ultimo numero del record)
oppure:
PRINT# numero del file, "P" CHR$ (numero del canale) CHR$ (primo numero del record) CHR$ (ultimo numero del record) CHR$ (posizione)

ESEMPI DEL COMANDO POSITION

PRINT# 15, "P" CHR$(2) CHR$(1) CHR$(0)
PRINT# 15, "P" CHR$(CH) CHR$(R1) CHR(R2)
PRINT# 15, "P" CHR$(4) CHR$(R1) CHR$(R2) CHR$(P)

È necessario usare il formato a 2 byte per il numero di record perché un byte può rappresentare fino a 256 numeri differenti, mentre possiamo avere in un file fino a 700 record. Il primo numero del record (low) contiene la parte meno significativa dell'indirizzo, mentre il secondo numero del record (high) costituisce la parte più significativa dello stesso. Il numero si ottiene con la formula:
Numero del record = REC HI * 256+REC LO

Supponiamo ora ad esempio di avere una lista di spedizione che contiene 8 tipi di dati, secondo il seguente schema:

Campo del nome	**Lunghezza del campo**
Nome	12
Cognome	15
Indirizzo (linea 1)	20
Indirizzo (linea 2)	20
Città	12
Stato	2
Codice Avviamento Postale	9
Numero telefonico	10
Totale	100

In questo modo è possibile determinare la lunghezza del record. Probabilmente avrete bisogno di un carattere extra in ogni campo per la separazione, altrimenti il comando INPUT# prenderebbe una parte di file più lunga di

quella effettivamente necessaria come nel caso di file sequenziali. Perciò, occorre considerare un file con 108 caratteri più record. Nel 1° record poniamo il numero 1 che rappresenta il più grande numero di record fin qui usato. Ecco il programma relativo:

```
 10 OPEN 1, 8, 15
 20 OPEN 2, 8, 3, "0:MAILING LIST,L,"+CHR$(108)
 30 GOSUB 900
 40 PRINT# 1, "p" CHR$(3) CHR$(1) CHR$(0) CHR$(1)
 50 GOSUB 900
 60 IF E=50 THEN PRINT#2, 1: GOTO 40
 70 INPUT# 2, X
300 STOP
900 INPUT# 1, E, B$, C, D
910 IF (E=50) OR (E<20) THEN RETURN
920 PRINT A; B; C; D: STOP: RETURN
```

L'errore numero 50 che può verificarsi alla linea 60 è l'errore di RECORD NOT PRESENT, il che significa che il record non è ancora stato creato. Scrivendo nel record, si risolve il problema. Questa condizione deve accuratamente essere controllata nell'ambito del vostro programma.

Fino a questo punto è stato creato il file ed il primo record, ma non sono ancora stati caricati dei dati in esso. Qui sotto è riportata una versione completa del programma che vi permette di lavorare con una lista di spedizione dove i record sono codificati per mezzo di numeri.

PROGRAMMA PER LA LETTURA E LA SCRITTURA DI UNA LISTA DI SPEDIZIONE

```
  5  A(1) = 12:A(2) = 15:A(3) = 20:A(4) = 20:A(5) = 12:A(6) = 2:A(7) = 9:A(8) = 10
 10  OPEN 1,8,15:OPEN2,8,3, "0:Mailing List,1,"+CHR$(108):GOSUB900
 20  PRINT#1,"p" CHR$(3) CHR$(1) CHR$(0) CHR$(1):PRINT#2,X
 30  INPUT"Read, Write, or End";J$:IFJ$="e"THENCLOSE2:CLOSE1:END
 40  IFJ$="w"THEN200
 50  PRINT:INPUT"Record" #";R:IFR<0ORR>XTHEN50
 60  IFR<2THEN30
 70  R1=R:R2=0:IFR1>256THENR2=INT(R1/256):R1=R1-256*R2
 80  RESTORE:DATI, FIRST NAME,14,LAST NAME,30,ADDRESS1,51,ADDRESS2
 90  DATA72, CITY,85,STATE,88,ZIP,98,PHONE#
100  FORL=1TO8:READA,A$:PRINT#1,"p"CHR$(13) CHR$(R1) CHR$(R2) CHR$(A):GOSUB900
110  ONA/50GOTO50:INPUT#2,Z$:PRINTA$,Z$:NEXT:GOTO50
200  PRINT:INPUT"Record #";R:IFR<0ORR>5000THEN200
210  IFR<2THEN30
215  IFR>XTHENR=W+1:PRINT:PRINT"Record #"R
220  R1=R:R2=0:IFR1>256THENR2=INT(R1/256):R1=R1 - 256*R2
230  RESTORE:FORL=1TO8:READA,A$:PRINT#1, "p"CHR$(3) CHR$(R1) CHR$(R2) CHR$(A)
240  PRINTA#,:INPUTZ$:IFLEN(Z$)>A(L)THENZ$=LEFT$(Z$,A(L))
245  PRINT#2,Z$:NEXT:X=PRINT#1,"p"CHR$(3) CHR$(1) CHR$(0)
250  PRINT#2,X:GOTO200
900  INPUT#1,A,B$,C,D:IFA<20THENRETURN
910  IFA<>=50THENPRINTA;B$.C;D:STOP:RETURN
920  IFJ$="r"THENPRINTB$
930  RETURN
```

Questo programma chiede i numeri dei record durante la loro ricerca. Ovviamente non vi permetterà di cercare oltre la fine del file e se vorrete scrivere oltre il limite, esso vi forzerà a scrivere nel record immediatamente successivo.

Una versione del programma più avanzata di questa dovrebbe tenere conto dei vari campi per mezzo di chiavi, in modo da ottenere un indice dei record. Ad esempio, voi vorrete probabilmente effettuare una ricerca per nome, o stampare delle etichette per CAP e quindi avete bisogno di un elenco separato di chiavi e di numeri dei record, memorizzate in un file sequenziale.

Quando dovete lavorare con un nuovo file relativo e pensate che esso assuma dimensioni notevoli, potete risparmiare tempo creando subito un record alla fine del file. Se ad esempio pensate che il file conterà 1000 record, è bene creare subito il record numero 1000 ed il DOS viene forzato a creare a sua volta tutti i record intermedi, permettendovi di lavorare molto velocemente.

ESEMPIO PER LA CREAZIONE DI UN FILE DI GRANDI DIMENSIONI

```
OPEN 1, 8, 15:OPEN 2, 8,2, "REL,L"+ CHR$(60)
PRINT# 1, "P" CHR$(2) CHR$(0) CHR$(4) CHR$(1)
PRINT# 2, "END"
CLOSE 2: CLOSE 1
```

8. **PROGRAMMAZIONE DEL CONTROLLORE DELL'UNITÀ**

I programmatori più esperti possono progettare delle routine residenti che operano sul controllore dell'unità a floppy disk. Le routine DOS possono essere aggiunte in memoria richiamandole dal floppy, nello stesso modo in cui sono aggiunte le abbreviazioni del programma DOS SUPPORT.

BLOCK-EXECUTE

Questo comando carica dal floppy in memoria un blocco contenente una routine in linguaggio macchina e ne lancia l'esecuzione dalla locazione 0 del buffer, fino a quando non viene incontrato un comando RTS (Return from Subroutine).

FORMATO DEL COMANDO **BLOCK-EXECUTE**

PRINT# numero del file, "BLOCK-EXECUTE:" canale, drive, traccia, blocco
o, in forma abbreviata:
PRINT# numero del file, "B-E:" canale, drive, traccia, blocco

MEMORY-READ

Nell'unità a floppy disk vi sono 16 k di ROM e 2 k di RAM. Potete accedere direttamente ad essi od ai buffer che il DOS ha inizializzato sulla RAM usando comandi MEMORY. MEMORY-READ vi permette di scegliere il byte che volete leggere per mezzo del **canale di errore.**

FORMATO PER IL COMANDO **MEMORY-READ**

PRINT #numero del file, "M-R:" CHR$ (1° byte dell'indirizzo) CHR$ (ultimo byte dell'indirizzo)
(senza abbreviazioni)

Il successivo byte letto con GET# sul canale 15 di errore, giunge dalla memoria del controllore del drive, all'indirizzo indicato ed i byte successivi, dalle locazioni seguenti.

Qualunque INPUT# al canale di errore vi darà dei risultati particolari, mentre usate questo comando. Il suo effetto può essere cancellato da un qualunque altro comando all'unità (oltre che da un comando MEMORY).

PROGRAMMA PER LEGGERE NELLA MEMORIA DEL CONTROLLORE DELL'UNITÀ A FLOPPY

```
 10 OPEN 15, 8, 15
 20 INPUT "LOCATION PLEASE"; A
 30 A1= INT(A/256): A2= A- A1*256
 40 PRINT# 15, "M-R:" CHR$(A2) CHR$(A1)
 50 FOR L=1 TO 5
 60 GET# 15, A$
 70 PRINT ASC(A$+ CHR$(0));
 80 NEXT
 90 INPUT "CONTINUE";A$
100 IF LEFT$(A$,1)="Y" THEN 50
110 GOTO 20
```

MEMORY-WRITE

Il comando MEMORY-WRITE vi permette di scrivere nella memoria del controllore fino a 34 byte in una sola volta. I comandi MEMORY-EXECUTE e USER permettono di lanciare il codice programma inserito.

FORMATO PER IL COMANDO **MEMORY-WRITE**

PRINT# numero del file. "M-W:" CHR$ (1° byte di indirizzo) CHR$ (ultimo byte di indirizzo) numero dei caratteri; dati

PROGRAMMA PER SCRIVERE UNA "RTS" SULL'UNITÀ

```
10 OPEN 15, 8, 15, "M-W:" CHR$(0) CHR$(5); 1; CHR$(96)
20 PRINT# 15, "M-E:" CHR$(0) CHR$(19): REM JUMPS TO BYTE, RETURNS
30 CLOSE 15
```

MEMORY-EXECUTE

Qualunque routine nella memoria DOS, RAM o ROM, può essere eseguita con il comando MEMORY-EXECUTE.
FORMATO PER IL COMANDO **MEMORY-EXECUTE**

PRINT# numero dei file, "M-E:" CHR$ (1° byte di indirizzo) CHR$ (ultimo byte di indirizzo)

Vedere, per esempio la precedente linea 20.

Comandi USER

Oltre ai comandi USER1 ed USER2 già visti nel Capitolo 6, ed ai comandi UI+ ed UI— al Capitolo 2, esistono altri comandi USER per effettuare salti a locazioni nella memoria RAM del controllore dell'unità a floppy.

COMANDI USER	FUNZIONE
U1 o UA	BLOCK-READ senza variazioni del puntatore di buffer
U2 o UB	BLOCK-WRITE senza variazioni del puntatore di buffer
U3 o UC	salta a $0500
U4 o UD	salta a $0503
U5 o UE	salta a $0506
U6 o UF	salta a $0509
U7 o UG	salta a $050C
U8 o UH	salta a $050F
U9 o UI	salta a $FFFA
U; o UJ	vettore power-up
UI+	configura l'unità per il Commodore 64
UI-	configura l'unità per il VIC 20

Caricando in queste locazioni altre istruzioni di salto come ad esempio JMP $0520, è possibile costruire lunghe routine che operano nella memoria dell'unità a disco con una tavola di salto facile da usare, perfino con il BASIC.

ESEMPI DI COMANDI **USER**

```
PRINT# 15, "U3"
PRINT# 15, "U"+ CHR$(50+Q)
PRINT# 15, "UI"
```

9. CAMBIAMENTO DEL NUMERO DI DISPOSITIVO DELL'UNITÀ

Per mezzo del software

Il numero del dispositivo viene selezionato dall'unità a floppy verificando la presenza di due ponticelli sul circuito stampato e scrivendo in RAM un valore basato sulla posizione di tali ponticelli. È possibile comunque cambiare via software tale numero.

FORMATO PER IL CAMBIAMENTO DEL NUMERO DI DISPOSITIVO

PRINT# numero del file, "M-W:" CHR$(119) CHR$(0) CHR$(2) CHR$(indirizzo +32) CHR$(indirizzo+64)

ESEMPIO DI CAMBIAMENTO DEL NUMERO DI DISPOSITIVO

PRINT# 15, "M-W:" CHR$(119) CHR$(0) CHR$(2) CHR$(9+32) CHR$(9+64)
PRINT# Q, "M-W:" CHR$(119) CHR$(0) CHR$(2) CHR$(R+32) CHR$ (R+64)

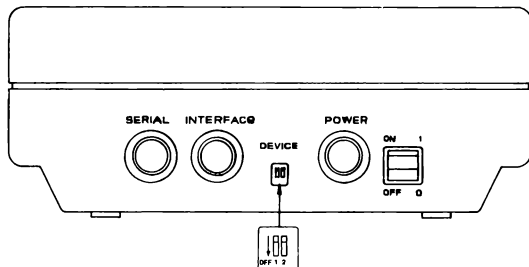
Se state lavorando con più di una unità a floppy, è meglio cambiare gli indirizzi intervenendo sull'hardware, come mostrato sotto, in modo semplice.

Per mezzo dell'hardware

Come modificare il numero identificatore di unità usando lo switch:

1. Spegnere il Disk Drive.
2. Fate riferimento alla tabella seguente per posizionare il DIP-SWITCH sul retro del drive per modificare il numero identificatore di unità:

Device#	SW1	SW2
8	ON	ON
9	OFF	ON
10	ON	OFF
11	OFF	OFF

3. Accendere il Disk Drive.

APPENDICE A: elenco dei comandi dell'unità a floppy

Formato generale: PRINT# numero del file, comando

COMANDO	FORMATO DEL COMANDO
NEW	"N
COPY	"CO: nuovo file=0: file originale
RENAME	"RO: nuovo file=0: vecchio file
SCRATCH	"SO: nome del file
INITIALIZE	"I
VALIDATE	"V
DUPLICATE	solo per drive multipli
BLOCK-READ	"B-R:" canale, drive, traccia, blocco
BLOCK-WRITE	"B-W:" canale, drive, traccia, blocco
BLOCK-ALLOCATE	"B-A:" drive, traccia, blocco
BLOCK-FREE	"B-F:" drive, traccia, blocco
BUFFER-POINTER	"B-P:" canale, posizione
USER1 e USER2	"Un:" canale, drive, traccia, blocco
POSITION	"P" CHR$ (numero di canale) CHR$ (1° byte di record CHR$ (ultimo byte di record) CHR$ (posizione)
BLOCK-EXECUTE	"B-E:" canale, drive, traccia, blocco
MEMORY-READ	"M-R:" CHR$ (1° byte di indirizzo) CHR$ (ultimo byte di indirizzo)
MEMORY-WRITE	"M-W:" CHR$ (1° byte di indirizzo) CHR$ (ultimo byte di indirizzo) CHR$ (numero del carattere) "dati"
MEMORY-EXECUTE	"M-E:" CHR$ (1° byte di indirizzo) CHR$ (ultimo byte di indirizzo
Comandi USER	"Un:"

APPENDICE B: Elenco dei messaggi di errore dell'unità a floppy CBM

0	Tutto OK. Non vi sono errori
1	Messaggio in risposta alla cancellazione di file. Non vi sono errori
2-19	Non usati come messaggio errore. Da ignorare
20	Intestazione del blocco non trovata sul disco
21	Mark di sincronismo non trovato
22	Blocco dati non presente
23	Errore di checksum nei dati
24	Errore di codifica del byte
25	Errore di verifica in scrittura
26	Tentativo di scrittura con protezione inserita
27	Errore di checksum nell'intestazione
28	Estensione dei dati nel blocco successivo
29	Mancata coincidenza dell'ID del disco
30	Errore generico di sintassi
31	Comando non valido
32	Linea troppo lunga
33	Nome di file non valido
34	Nessun file è dato
39	File comando non trovato
50	Record non presente
51	Overflow su record
52	File troppo grande
60	File aperto per scrittura
61	File non aperto
62	File non trovato
63	File già esistente
64	Verifica fallita per il tipo di file
65	Nessun blocco
66	Traccia o settore illegale
67	Traccia o settore illegale del sistema
70	Nessun canale disponibile
71	Errore nel direttorio
72	Disco o direttorio completo
73	Messaggio di accensione o tentativo di scrittura con versione DOS non compatibile
74	Unità a disco non pronta (solo con 8050)

DESCRIZIONE DEI MESSAGGI DI ERRORE DOS

NOTA: i messaggi di errore con numero inferiore a 20 debbono essere ignorati, a parte lo 01 che dà informazioni sul numero di file cancellati con il comando SCRATCH

20: READ ERROR (intestazione del blocco non trovata)
Il controllore non riesce a trovare l'intestazione del blocco dati cercato. L'errore è causato da un numero di settore non valido o l'intestazione del blocco è stata distrutta.

21: READ ERROR (Mark di sincronismo non trovato)
Il controllore non è in grado di trovare il mark di sincronismo sulla traccia desiderata. L'errore può essere causato da un disallineamento della testina di lettura/scrittura, dalla mancanza del floppy nell'unità o da un floppy inserito male. Può anche indicare un guasto hardware.

22: READ ERROR (Blocco dati non presente)
Al controllore è stato chiesto di leggere o verificare un blocco dati che non è stato scritto correttamente. Questo messaggio di errore si verifica sempre con i comandi BLOCK ed indica una richiesta di traccia e/o settore illegali.

23: READ ERROR: (Blocco di checksum nei dati)
Esiste un errore in uno o più byte di dati. Il dato è stato letto nella memoria DOS, ma esiste un errore di checksum. Il messaggio può anche indicare problemi di messa a terra.

24: READ ERROR (Errore di decodifica del byte)
I dati o l'intestazione sono stati letti nella memoria del DOS, ma si è verificato un errore hardware dovuto ad una sequenza non valida di bit nei byte di dati. Questo messaggio può anche indicare dei problemi di messa a terra.

25: WRITE ERROR (Errore di verifica in scrittura)
Questo messaggio viene generato quando il controllore rivela una non coincidenza tra i dati scritti ed i dati dati presenti nella memoria del DOS.

26: WRITE PROTECTION
È stato chiesto al controllore di scrivere un blocco di dati mentre il tasto di protezione a scrittura è attivato. Normalmente è causato dalla presenza della linguetta di protezione sul floppy.

27: READ ERROR (Errore di checksum nell'intestazione)
Il controllore ha rivelato un errore nell'intestazione del blocco deti richiesto. Il blocco non è stato letto nella memoria del DOS. Il messaggio può anche indicare dei problemi di messa a terra.

28: WRITE ERROR (Blocco dati troppo lungo)
Il controllore cerca di trovare un mark di sincronismo dell'intestazione successiva dopo aver scritto un blocco di dati; se il mark non compare entro un tempo fissato, viene generato il messaggio di errore. L'errore può essere causato da un floppy formattato male (i dati si estendono sul blocco successivo) o da un guasto hardware.

29: DISK ID MISMATCH
È stato chiesto al controllore di accedere ad un floppy che non è stato inizializzato. Il messaggio si può anche verificare per cattiva intestazione del floppy.

30: SYNTAX ERROR (Generico)
Il DOS non riesce ad interpretare il comando inviato. Generalmente si verifica per un numero di file illegale od una sequenza usata in modo illegale. Ad esempio, due nomi di file a sinistra del comando COPY.

31: SYNTAX ERROR (Comando non valido)
Il DOS non riconosce il comando. I comandi debbono partire in prima posizione.

32: SYNTAX ERROR (Linea troppo lunga)
Il comando inviato è più lungo di 58 caratteri.

33: SYNTAX ERROR (Nome del file non valido)
L'accoppiamento di sequenze è usato in modo illegale con i comandi OPEN e SAVE.

34: SYNTAX ERROR (Nessun file viene fornito)
Il nome del file manca nel comando o il DOS non lo riconosce come tale. Generalmente è stato dimenticato (:).

39: SYNTAX ERROR (Comando non valido)
Il comando inviato non viene riconosciuto dal DOS

50: RECORD NOT PRESENT
Risulta dal tentativo di lettura su un disco passato l'ultimo record con INPUT# o GET#.

51: OVERFLOW IN RECORD
PRINT# eccede i limiti del record. L'informazione viene troncata. Poiché il ritorno del carrello inviato come terminatore del record viene conteggiato nelle dimensioni dello stesso, il messaggio si verifica se i caratteri totali del record, compreso CR, superano tale dimensione.

52: FILE TOO LARGE
La posizione del record all'interno di un file relativo indica un overflow del disco.

60: WRITE FILE OPEN
Un file di scrittura che non è stato chiuso viene aperto in lettura.

61: FILE NOT OPEN
Si cerca di accedere ad un file che non è stato aperto dal DOS. Talvolta, non viene generato il messaggio e la richiesta è semplicemente ignorata.

62: FILE NOT FOUND
Il file cercato non esiste sul floppy in uso.

63: FILE EXISTS
Il file che si cerca di creare esiste già sul floppy in uso.

64: FILE TYPE MISMATCH
Il tipo di file non è lo stesso di quello presente nel direttorio alla voce del file cercato.

65: NO BLOCK
Si verifica con il comando B-A. Indica che il blocco che si desidera allocare è già stato allocato. I parametri indicano la traccia ed il settore immediatamente successivo disponibile. Se sono pari a 0, tutti i blocchi con numero maggiore sono già stati usati.

66: ILLEGAL TRACK AND SECTOR
Il DOS ha cercato di accedere ad una traccia o ad un settore che non esiste nel formato in uso. Ciò può indicare un errore di lettura per il puntatore del blocco successivo.

67: ILLEGAL SYSTEM T OR S
Indica settore o traccia illegali di sistema.

70: NO CHANNEL (disponibile)
Il canale richiesto non è disponibile, o tutti i canali sono già in uso. Con il DOS possono essere aperti al massimo 5 file sequenziali contemporanei. I canali ad accesso diretto possono avere 6 file aperti contemporaneamente.

71: DIRECTORY ERROR
Il BAM non riesce a verificare esatto il conteggio interno. Si tratta di un problema di allocazione del BAM o lo stesso è stato cancellato e riscritto nella memoria del DOS. Per superare cio, occorre riinizializzare il floppy e ristrutturare il BAM. Alcuni file attivi possono essere terminati dall'azione correttiva.

72: DISK FULL
O tutti i blocchi del disco sono stati usati, oppure il direttorio ha raggiunto i limiti di 144 voci per il 1451, di 152 per 2040, 3040 e 4040 o di 243 per 8050. DISK FULL viene inviato sull'8050 quando vi sono ancora due blocchi disponibili per permettere di chiudere il file corrente.

73: DOS MISMATCH (73, CBM DOS V 2.6 1541)
I DOS versione 1 e 2 sono compatibili in lettura, ma non in scrittura, poiché i formati sono differenti. L'errore si verifica quando si cerca di scrivere su un floppy formattato con un formato non compatibile (esiste una routine di utilità per la conversione tra i formati). Il messaggio può anche apparire al momento dell'accesnsione.

74: DRIVE NOT READY
Si è cercato di accedere all'unità 1541 senza floppy disk inserito nella stessa.

APPENDICE C: Programmi contenuti nel floppy disk dimostrativo

1. DIR

```
4 OPEN2,8,15
5 PRINT"□":GOTO 10000
10 OPEN1,8,0,"$0"
20 GET#1,A$,B$
30 GET#1,A$,B$
40 GET#1,A$,B$
50 C=0
60 IF A$<>"" THEN C=ASC(A$)
70 IF B$<>"" THEN C=C+ASC(B$)*256
80 PRINT"■"MID$(STR$(C),2);TAB(3);"■";
90 GET#1,B$:IF ST<>0 THEN 1000
100 IF B$<>CHR$(34) THEN 90
110 GET#1,B$:IF B$<>CHR$(34)THEN PRINTB$;:GOTO110
120 GET#1,B$:IF B$=CHR$(32) THEN 120
130 PRINT TAB(18);:C$=""
140 C$=C$+B$:GET#1,B$:IF B$<>"" THEN 140
150 PRINT"■"LEFT$(C$,3)
160 GET T$:IF T$<>"" THEN GOSUB 2000
170 IF ST=0 THEN 30
1000 PRINT" BLOCKS FREE"
1010 CLOSE1:GOTO 10000
2000 IF T$="Q" THEN CLOSE1:END
2010 GET T$:IF T$="" THEN 2000
2020 RETURN
4000 REM DISK COMMAND
4010 C$="":PRINT">";
4011 GETB$:IFB$="" THEN4011
4012 PRINTB$;:IF B$=CHR$(13) THEN 4020
4013 C$=C$+B$:GOTO 4011
4020 PRINT#2,C$
5000 PRINT"■";
5010 GET#2,A$:PRINTA$;:IF A$<>CHR$(13)GOTO5010
5020 PRINT"■"
10000 PRINT "D-DIRECTORY"
10010 PRINT ">-DISK COMMAND"
10020 PRINT "Q-QUIT PROGRAM"
10030 PRINT "S-DISK STATUS "
10100 GETA$:IFA$=""THEN10100
10200 IF A$="D" THEN 10
10300 IF A$="." OR A$=">" OR A$=">" THEN 4000
10310 IF A$="Q" THEN END
10320 IF A$="S" THEN 5000
10999 GOTO 10100
```

2. VIEW BAM

```
100 REM ********************************
101 REM *  VIEW BAM FOR VIC & 64 DISK  *
102 REM ********************************
105 OPEN15,8,15
110 PRINT#15,"I0":NU$="N/A N/A N/A N/A N/A":Z4=1
120 OPEN2,8,2,"#"
130 Y$="■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■"
140 X$="■■■■■■■■■■■■■■■■■■■■■■■■■"
150 DEF FNS(Z) = 2↑(S-INT(S/8)*8) AND (SB(INT(S/8)))
```

```
160 PRINT#15,"U1:";2;0;18;0
170 PRINT#15,"B-P";2;1
180 PRINT"■";
190 Y=22:X=1:GOSUB430
200 FORI=0TO20:PRINT:PRINT"■■"RIGHT$(STR$(I)+"  ",3);:NEXT
210 GET#2,A$
220 GET#2,A$
230 GET#2,A$
240 TS=0
250 FORT=1TO17:GOSUB450
260 Y=22:X=T+4:GOSUB430:GOSUB540:NEXT
270 FORI=1TO2000:NEXT:PRINT"■"
280 Y=22:X=1:GOSUB430
290 FORI=0TO20:PRINT:PRINT"■■"RIGHT$(STR$(I)+"  ",3);:NEXT
300 FORT=18TO35
310 GOSUB450
320 Y=22:X=T-13:GOSUB430:GOSUB540:NEXT
330 FORI=1TO1000:NEXT
340 PRINT"■■■■■■■"
350 PRINT#15,"B-P";2;144
360 N$="":FORI=1TO20:GET#2,A$:N$=N$+A$:NEXT
370 PRINT"  "N$"  "TS-17;"BLOCKS FREE"
380 FORI=1TO4000:NEXT
390 PRINT"■"
400 INPUT"■■■■■ANOTHER DISKETTE N■■■■";A$
410 IFA$="Y"THENRUN
420 IFA$<>"Y"THENEND
430 PRINTLEFT$(Y$,Y)LEFT$(X$,X)"■■";
440 RETURN
450 GET#2,SC$:SC=ASC(RIGHT$(CHR$(0)+SC$,1))
460 TS=TS+SC
470 GET#2,A$:IFA$=""THENA$=CHR$(0)
480 SB(0)=ASC(A$)
490 GET#2,A$:IFA$=""THENA$=CHR$(0)
500 SB(1)=ASC(A$)
510 GET#2,A$:IFA$=""THENA$=CHR$(0)
520 SB(2)=ASC(A$)
530 RETURN
540 PRINT"■■"RIGHT$(STR$(T),1);"■■■";
550 REM  PRINTT"    "SC"   "SB(0)" "SB(1)"  "SB(2)=CHR$(0)
560 IFT>24ANDS=18THEN:PRINTMID$(NU$,Z4,1);:GOTO660
570 FORS=0TO20
580 IFT<18THEN620
590 IFT>30ANDS=17THEN:PRINTMID$(NU$,Z4,1);:GOTO660
600 IFT>24ANDS=18THEN:PRINTMID$(NU$,Z4,1);:GOTO660
610 IFT>24ANDS=19THENPRINTMID$(NU$,Z4,1);:GOTO660
620 IFT>17ANDS=20THENPRINTMID$(NU$,Z4,1);:Z4=Z4+1:GOTO660
630 PRINT"■";
640 IF FNS(S)=0 THEN PRINT"+";:GOTO660
650 PRINT"■+";:REMRIGHT$(STR$(S),1);Z4,1);:GOTO72
660 PRINT"■■■";
670 NEXT
680 RETURN
```

3. DISPLAY T&S

```
100 REM********************************
110 REM* DISPLAY ANY TRACK $ SECTOR *
120 REM* ON THE DISK TO THE SCREEN  *
130 REM* OR THE PRINTER             *
140 REM********************************
150 PRINT"■■■____________________________"
160 PRINT"DISPLAY BLOCK CONTENTS"
165 PRINT"‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾":
170 REM********************************
180 REM* SET PROGRAM CONSTANT       *
190 REM********************************
200 SP$=" ":NL$=CHR$(0):HX$="0123456789ABCDEF"
210 FS$="":FORI=64 TO 95:FS$=FS$+"■"+CHR$(I)+"■":NEXT I
220 SS$="  ":FOR I=192 TO 223:SS$=SS$+"■"+CHR$(I)+"■":NEXT I
240 DIM A$(15),NB(2)
251 D$="0"
253 PRINT"            ■S■CREEN■■■■■■■■■■■OR ■■■P■RINTER"
254 GETJJ$:IF JJ$="" THEN254
255 IF JJ$="S"THENPRINT"          ■■SCREEN■"
256 IF JJ$="P"THENPRINT"          ■■PRINTER■"
260 OPEN15,8,15,"I"+D$:GOSUB 650
265 OPEN4,4
270 OPEN 2,8,2,"#":GOSUB 650
280 REM********************************
290 REM* LOAD TRACK AND SECTOR      *
300 REM* INTO DISK BUFFER           *
310 REM********************************
320 INPUT"■■■TRACK, SECTOR";T,S
330 IF T=0 OR T>35 THEN PRINT#15;"I"D$:CLOSE2:CLOSE4:CLOSE15:PRINT"END":END
340 IF JJ$="S" THEN PRINT"■■■TRACK"T" SECTOR"S"■"
341 IF JJ$="P" THEN PRINT#4:PRINT#4,"TRACK"T" SECTOR"S:PRINT#4
350 PRINT#15,"U1:2,"D$;T;S:GOSUB650
360 REM********************************
370 REM* READ BYTE 0 OF DISK BUFFER *
390 REM********************************
400 PRINT#15,"B-P:2,1"
410 PRINT#15,"M-R"CHR$(0)CHR$(5)
420 GET#15,A$(0):IFA$(0)=""THENA$(0)=NL$
428 IF JJ$="S"THEN430
430 IF JJ$="P"THEN460
431 REM********************************
432 REM* READ & CRT DISPLAY         *
433 REM* REST OF THE DISK BUFFER    *
434 REM********************************
436 K=1:NB(1)=ASC(A$(0))
438 FOR J=0 TO 63:IF J=32 THEN GOSUB 710:IF Z$="N"THEN J=80:GOTO 458
440 FOR I=K TO 3
442 GET#2,A$(I):IF A$(I)="" THEN A$(I)=NL$
444 IF K=1 AND I<2 THEN NB(2)=ASC(A$(I))
446 NEXT I:K=0
448 A$="":B$=":":N=J*4:GOSUB 790:A$=A$+":"
450 FOR I=0 TO 3:N=ASC(A$(I)):GOSUB 790
452 C$=A$(I):GOSUB 850:B$=B$+C$
454 NEXT I:IF JJ$="S" THEN PRINTA$B$
458 NEXT J:GOTO571
```

```
460 REM*********************************
462 REM* READ & PRINTER DISPLAY        *
464 REM*********************************
466 K=1:NB(1)=ASC(A$(0))
468 FOR J=0 TO 15
470 FOR I=K TO 15
472 GET#2,A$(I):IF A$(I)="" THEN A$(I)=NL$
474 IF K=1 AND I<2 THEN NB(2)=ASC(A$(I))
476 NEXT I:K=0
478 A$="":B$=":":N=J*16:GOSUB 790:A$=A$+":"
480 FOR I=0 TO 15:N=ASC(A$(I)):GOSUB 790:IF Z$="N"THEN J=40:GOTO 571
482 C$=A$(I):GOSUB 850:B$=B$+C$
484 NEXT I
486 IF JJ$="P" THEN PRINT#4,A$B$
488 NEXT J:GOTO571
571 REM*********************************
572 REM* NEXT TRACK AND SECTOR         *
573 REM*********************************
575 PRINT"NEXT TRACK AND SECTOR"NB(1)NB(2) "■"
580 PRINT"DO YOU WANT NEXT TRACK AND SECTOR"
590 GET Z$:IF Z$="" THEN590
600 IF Z$="Y" THEN T=NB(1):S=NB(2):GOTO330
610 IF Z$="N" THEN 320
620 GOTO 590
630 REM*********************************
640 REM* SUBROUTINES                   *
650 REM*********************************
660 REM* ERROR ROUTINE                 *
670 REM*********************************
680 INPUT#15,EN,EM$,ET,ES:IF EN=0 THEN RETURN
690 PRINT"■DISK ERROR■"EN,EM$,ET,ES
700 END
710 REM*********************************
720 REM* SCREEN CONTINUE MESSAGE       *
730 REM*********************************
740 PRINT"■■■■■CONTINUE(Y/N)"
750 GETZ$:IF Z$="" THEN 750
760 IF Z$="N" THEN RETURN
770 IF Z$<>"Y" THEN 750
780 PRINT"■TRACK" T " SECTOR"S "■":RETURN
790 REM*********************************
800 REM* DISK BYTE TO HEX PRINT        *
810 REM*********************************
820 A1=INT(N/16):A$=A$+MID$(HX$,A1+1,1)
830 A2=INT(N-16*A1):A$=A$+MID$(HX$,A2+1,1)
840 A$=A$+SP$:RETURN
850 REM*********************************
860 REM* DISK BYTE TO ASC DISPLAY      *
870 REM* CHARACTER                     *
880 REM*********************************
890 IF ASC(C$)<32 THEN C$=" ":RETURN
910 IF ASC(C$)<128 OR ASC(C$)>159 THEN RETURN
920 C$=MID$(SS$,3*(ASC(C$)-127),3):RETURN
```

4. CHECK DISK

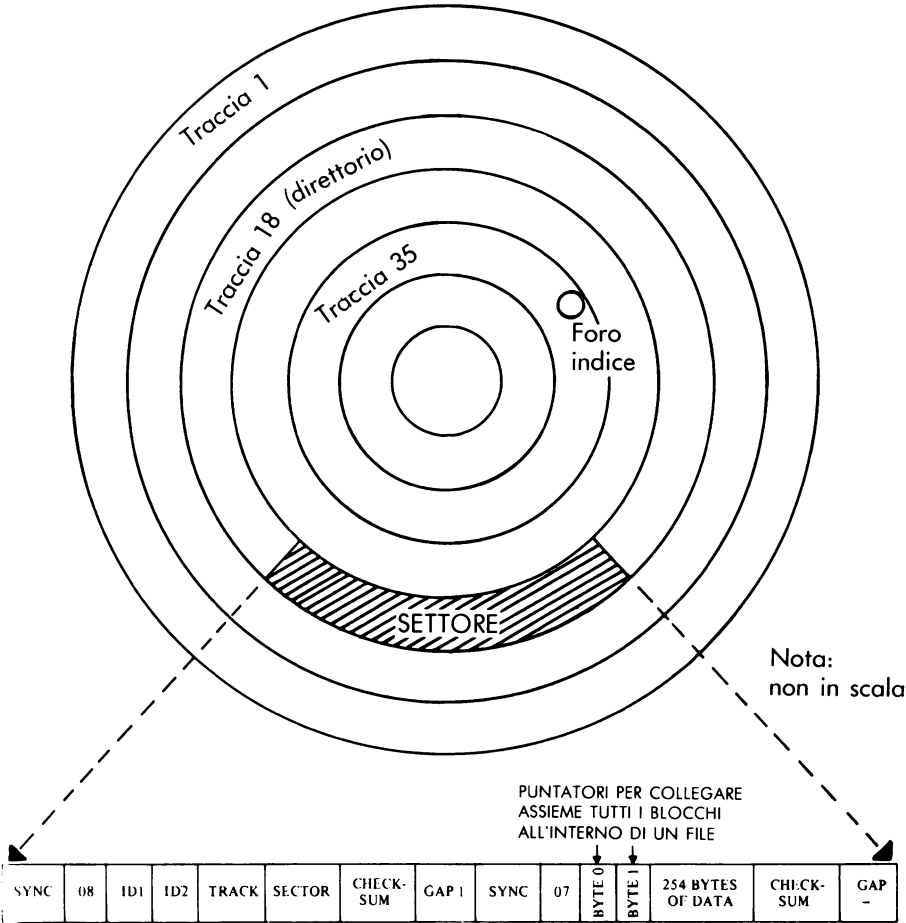
```
1 REM CHECK DISK -- VER 1.4
2 DN=8:REM FLOPPY DEVICE NUMBER
5 DIMT(100):DIMS(100):REM BAD TRACK, SECTOR ARRAY
9 PRINT"█████─────────────────────────"
10 PRINT" CHECK DISK PROGRAM"
12 PRINT"──────────────────────────"
20 D$="0"
30 OPEN15,DN,15
35 PRINT#15,"V"D$
45 N%=RND(TI)*255
50 A$="":FORI=1TO255:A$=A$+CHR$(255AND(I+N%)):NEXT
60 GOSUB900
70 OPEN2,DN,2,"#"
80 PRINT:PRINT#2,A$;
85 T=1:S=0
90 PRINT#15,"B-A:"D$;T;S
100 INPUT#15,EN,EM$,ET,ES
110 IFEN=0THEN130
115 IFET=0THEN200:REM END
120 PRINT#15,"B-A:"D$;ET;ES:T=ET:S=ES
130 PRINT#15,"U2:2,"D$;T;S
134 NB=NB+1:PRINT" CHECKED  BLOCKS"NB
135 PRINT" TRACK    █████"T;" SECTOR    █████"S"██"
140 INPUT#15,EN,EM$,ES,ET
150 IF EN=0THEN85
160 T(J)=T:S(J)=S:J=J+1
165 PRINT"███BAD BLOCK:███",T;S""
170 GOTO85
200 PRINT#15,"I"D$
210 GOSUB900
211 PRINT#15,"V"
212 CLOSE2
215 IFJ=0THENPRINT"███████NO BAD BLOCKS!":END
217 OPEN2,DN,2,"#"
218 PRINT"███BAD BLOCKS","TRACK","SECTOR"
220 FORI=0TOJ-1
230 PRINT#15,"B-A:";D$,T(I);S(I)
240 PRINT,,T(I),S(I)
250 NEXT
260 PRINT"█"J"BAD BLOCKS HAVE BEEN ALLOCATED"
270 CLOSE2:END
900 INPUT#15,EN,EM$,ET,ES
910 IF EN=0 THEN RETURN
920 PRINT"███ERROR #"EN,EM$;ET;ES""
930 PRINT#15,"I"D$

READY.
```

5. PERFORMANCE TEST

```
1000 REM  PERFORMANCE TEST  2.0
1010 :
1020 REM  VIC-20 AND COMMODORE 64
1030 REM  SINGLE FLOPPY DISK DRIVE
1040 :
1050 OPEN 1,8,15:OPEN15,8,15
1060 LT=35
1070 LT$=STR$(LT)
```

```
1080 NT=30
1090 PRINT"⌂▮────────────────────────"
1100 PRINT"   PERFORMANCE TEST"
1110 PRINT"──────────────────────"
1120 PRINT
1130 PRINT"  INSERT SCRATCH"
1140 PRINT
1150 PRINT"    DISKETTE IN DRIVE"
1160 PRINT
1170 PRINT"▮   PRESS ▮RETURN▮"
1180 PRINT
1190 PRINT"            WHEN READY▮"
1200 FOR I=0 TO 50:GET A$:NEXT
1210 GET A$:IF A$<>CHR$(13) THEN 1210
1220 :
1230 :
1240 TI$="000000"
1250 TT=18
1260 PRINT#1,"N0:TEST DISK,00"
1270 C1$="   DISK NEW COMMAND   "+CHR$(13)
1280 C2$="▮ WAIT ABOUT 80 SECONDS"
1290 CC$=C1$+C2$:GOSUB 1840
1300 IF TI<NTTHEN1370
1310 PRINT"▮SYSTEM IS"
1320 PRINT"▮          NOT RESPONDING"
1330 PRINT" CORRECTLY TO COMMANDS"
1340 GOSUB 1880
1350 :
1360 :
1370 PRINT"▮DRIVE PASS"
1380 PRINT"          MECHANICAL TEST▮"
1390 TT=21
1400 OPEN 2,8,2,"0:TEST FILE,S,W"
1410 CC$="OPEN WRITE FILE"          :GOSUB 1840
1420 CH=2:CC$="WRITE DATA"          :GOSUB 1930
1430 CC$="CLOSE "+CC$               :GOSUB 1840
1440 OPEN 2,8,2,"0:TEST FILE,S,R"
1450 CC$="OPEN READ FILE"           :GOSUB 1840
1460 CH=2:GOSUB 1990
1470 PRINT#1,"S0:TEST FILE"
1480 CC$="SCRATCH FILE▮":TT=1       :GOSUB 1840
1490 :
1500 :
1510 TT=21
1520 OPEN 4,8,4,"#"
1530 NN%=(1+RND(TI)*254+NN%)AND255:PRINT#1,"B-P";4;NN%
1540 NN$="":FOR I=1 TO 255:NN$=NN$+CHR$(I):NEXT
1550 PRINT# 4,NN$;
1560 PRINT# 1,"U2:";4;0;LT;0
1570 CC$="WRITE TRACK"+LT$:GOSUB 1840
1580 PRINT#1,"U2:";4;0;1;0
1590 CC$="WRITE TRACK 1"            :GOSUB 1840
1600 PRINT#1,"U1:";4;0;LT;0
1610 CC$="READ TRACK"+LT$           :GOSUB 1840
1620 PRINT#1,"U1:";4;0;1;0
1630 CC$="READ TRACK 1"             :GOSUB 1840
1640 CLOSE 4
1650 :
1660 :
```

```
1670 PRINT"▮ UNIT HAS PASSED"
1680 PRINT"     PERFORMANCE TEST!"
1690 PRINT"▮ PULL DISKETTE FROM"
1700 PRINT"▮  DRIVE BEFORE TURNING"
1710 PRINT"   POWER OFF."
1720 END
1730 :
1740 :
1750 PRINT"   ▮CONTINUE (Y/N)?";
1760 FOR I=0 TO 50:GET A$:NEXT
1770 GET A$:IF A$="" THEN 1770
1780 PRINT A$"▮"
1790 IF A$="N" THEN END
1800 IF A$="Y" THEN RETURN
1810 GOTO 1760
1820 :
1830 :
1840 PRINT CC$
1850 INPUT# 1,EN,EM$,ET,ES
1860 PRINTTAB(12)""EN;EM$;ET;ES;""
1870 IF EN<2 THEN RETURN
1880 PRINT"▮ UNIT IS FAILING"
1890 PRINT"▮   PERFORMANCE TEST"
1900 TM$=TI$:GOSUB 1750:TI$=TM$:RETURN
1910 :
1920 :
1930 PRINT"WRITING DATA"
1940 FOR I=1000 TO 2000:PRINT#CH,I:NEXT
1950 GOSUB1850
1960 CLOSE CH:RETURN
1970 :
1980 :
1990 PRINT"READING DATA"
2000 GETA$
2010 FOR I=1000 TO 2000
2020 INPUT# CH,J
2030 IF J<>I THEN PRINT"▮READ ERROR:▮":GOSUB 1850
2040 NEXT
2050 GOSUB 1850
2060 CLOSE CH:RETURN
```

APPENDICE D: FORMATI SUL FLOPPY DISK

Formato del 1540/1541: espansione di un settore

Distribuzione dei blocchi per ogni traccia

Numeri di traccia sull'unità 2040, 3040	Campo dei blocchi o dei settori	Totale
da 1 a 17 da 18 a 24 da 25 a 30 da 31 a 35	da 0 a 20 da 0 a 19 da 0 a 17 da 0 a 16	21 20 18 17
Numeri di traccia sull'unità 4040	Campo dei blocchi o dei settori	Totale
da 1 a 17 da 18 a 24 da 25 a 30 da 31 a 35	da 0 a 20 da 0 a 18 da 0 a 17 da 0 a 16	21 19 18 17
Numeri di traccia sull'unità 8050	Campo dei blocchi o dei settori	Totale
da 1 a 39 da 40 a 53 da 54 a 64 da 65 a 77	da 0 a 28 da 0 a 26 da 0 a 24 da 0 a 22	29 27 25 23

FORMATO BAM DEL 1540/1541

Traccia 18, settore 0.		
BYTE	CONTENUTO	DEFINIZIONE
0,1	18,01	Traccia e settore del 1° blocco del direttorio
2	65	Codice ASCII del carattere A che indica il formato del 4040
3	0	Flag nullo per usi futuri del DOS
4 - 143		* bit map dei blocchi disponibili 1-35
*1 = blocco disponibile 0 = blocco non disponibile (ogni bit rappresenta un blocco)		

* STRUTTURA DI UN SINGOLO CAMPO DEL DIRETTORIO

BYTE	CONTENUTO	DEFINIZIONE
0	128+tipo	Tipo del file in OR logico con $80 per indicare correttamente la chiusura del file. TIPI: 0 = DELeted (cancellato) 1 = SEQuenziale 2 = PROGramma 3 = USER 4 = RELativo
1-2		Traccia e settore del 1° blocco di dati
3-18		Nome del file con spazi shiftati
19-20		Solo per file relativi: traccia e settore di inizio
21		Solo per file relativi: dimensioni del record
22-25		Non usato
26-27		Traccia e settore di sostituzione del file con OPEN @
28-29		Numero dei blocchi nel file: primo ed ultimo byte

FORMATO DI FILE SEQUENZIALE

BYTE	DEFINIZIONE
0,1	Traccia e settore del successivo blocco dati
2-256	254 byte di dati con ritorno carrello e terminatori del record

FORMATO DI FILE PROGRAMMA

BYTE	DEFINIZIONE
0,1	Traccia e settore del successivo blocco
2-256	254 byte di programma in formato CBM. La fine del file è individuata da 3 byte nulli.

TITOLO DEL DIRETTORIO DEL 1540/1541

Traccia 18, Settore 0		
BYTE	CONTENUTO	DEFINIZIONE
144-161		Nome del disco con spazi shiftati
162-163		ID del floppy
164	160	Spazio shiftato
165-166	50,65	Codice ASCII di 2A che rappresenta la versione ed il formato del DOS
166-167	160	Spazi shiftati
171-255	0	Nulli, non usati
Nota: codici ASCII possono comparire nelle locazioni da 180 a 191 di alcuni floppy.		

FORMATO DEL DIRETTORIO

Traccia 18, settore 1 per 4040	
Traccia 39, settore 1 per 8050	
BYTE	DEFINIZIONE
0,1	Traccia e settore del blocco successivo del direttorio
2-31	* Campo file 1
34-63	* Campo file 2
66-95	* Campo file 3
98-127	* Campo file 4
130-159	* Campo file 5
162-191	* Campo file 6
194-223	* Campo file 7
226-255	* Campo file 8

FORMATO DI FILE RELATIVI

BLOCCO DATI	
BYTE	DEFINIZIONE
0,1	Traccia e settore del successivo blocco di dati
2-256	254 byte di dati. I record vuoti contengono FF (tutti uni binari) nel primo byte, seguiti da 00 (tutti zeri binari) fino a completamento del record. I record parzialmente occupati sono riempiti con (00).
BLOCCO SIDE SECTOR	
BYTE	DEFINIZIONE
0,1	Traccia e settore del successivo blocco di dati
2	Numero del side sector (0-5)
3	Lunghezza del record
4-5	Traccia e settore del primo side sector (numero 0)
6-7	Traccia e settore del secondo side sector (numero 1)
8-9	Traccia e settore del terzo side sector (numero 2)
10-11	Traccia e settore del quarto side sector (numero 3)
12-13	Traccia e settore del quinto side sector (numero 4)
14-15	Traccia e settore del sesto side sector (numero 5)
16-256	Puntatori di traccia e settore ai 120 blocchi dati

BESCHEINIGUNG DES HERSTELLERS

Hiermit wird bestätigt, dass die Floppy Disk

COMMODORE 1541-II

in Übereinstimmung mit den Bestimmungen der

Amtsblattverfügung Nr. 1046/1984

funk-entstört ist.

Der Deutschen Bundespost wurde das Inverkehrbringen dieses Gerätes angezeigt und die Berechtigung zur Überprüfung der Serie auf Einhaltung der Bestimmungen eingeräumt.

COMMODORE BÜROMASCHINEN GMBH

CERTIFICATE OF THE MANUFACTURER

Herewith we certify that our device Floppy Disk

COMMODORE 1541-II

corresponds to the regulations

Amtsblattverfügung Nr. 1046/1984

is eliminated of radio interference.

The German Bundespost has been informed that this unit is on the market and has got the right to check on the mass production if the limits are kept.

COMMODORE BUSINESS MACHINES LIMITED